Brittle Features of Device Authentication

Washington Garcia
w.garcia@ufl.edu
University of Florida
Gainesville, Florida, USA

Animesh Chhotaray
chho58@ufl.edu
University of Florida
Gainesville, Florida, USA

Joseph I. Choi
choijoseph007@ufl.edu
University of Florida
Gainesville, Florida, USA

Suman Kalyan Adari
sadari@ufl.edu
University of Florida
Gainesville, Florida, USA

Kevin R.B. Butler
butler@ufl.edu
University of Florida
Gainesville, Florida, USA

Somesh Jha
jha@cs.wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

ABSTRACT

Authenticating a networked device relies on identifying its unique characteristics. Recent device fingerprinting proposals demonstrate that device activity, such as network traffic, can be used to extract features which identify devices using machine learning (ML). However, there has been little work examining how adversarial machine learning can compromise these schemes. In this work, we show two efficient attacks against three ML-based device authentication (MDA) systems. One of the attacks is an adaptation of an existing gradient-estimation-based attack to the MDA setting; the second uses a fuzzing-based approach. We find that the MDA systems use brittle features for device identification and hence, can be reliably fooled with only 30 to 80 failed authentication attempts. However, selecting features that are robust against adversarial attack is challenging, as indicators such as information gain are not reflective of the features that adversaries most profitably attack. We demonstrate that it is possible to defend MDA systems which rely on neural networks, and in the general case, offer targeted advice for designing more robust MDA systems.

CCS CONCEPTS

• **Security and privacy → Authentication**; • **Computing methodologies → Machine learning approaches**.

KEYWORDS

device fingerprinting, device authentication, adversarial machine learning, traffic analysis

ACM Reference Format:
Washington Garcia, Animesh Chhotaray, Joseph I. Choi, Suman Kalyan Adari, Kevin R.B. Butler, and Somesh Jha. 2021. Brittle Features of Device Authentication. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (CODASPY '21), April 26–28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3422337.3447842

CODASPY '21, April 26–28, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8143-7/21/04…$15.00
https://doi.org/10.1145/3422337.3447842

1 INTRODUCTION

Authentication is a classically difficult systems security problem, but is integral to the establishment of identity and ultimately trust in computer networks. In the era of the Internet of Things, which has seen an explosion in the number of wireless communication-aware devices, the problem of *device authentication* has become increasingly urgent. In these environments, it is advantageous to identify devices without extra modifications to their storage or communication requirements. Thus, methods adapted to leverage characteristics of the devices have been a focus of recent research (e.g., identifying a device based on the characteristics of its network traffic alone [12]). To facilitate this goal, machine learning (ML) has become a means of establishing a device's identity [4, 11, 40].

In an *ML-based device authentication* (MDA) system, an underlying ML model maps a set of device signatures to a set of credentials. However, there has been little consideration to date for the robustness of the underlying models in device authentication settings, particularly against iteratively-crafted adversarial samples. Due to different requirements and threat models, the device fingerprinting literature contains many implementations and signature extraction methods that span the hardware and software stack. Likewise, there is not yet a unified standard for performing attestation of network devices using signatures. Our goal is to investigate the feasibility of model-based device fingerprinting as a medium for performing the device authentication task.

The field of adversarial machine learning (AML) has uncovered several milestone attacks against state-of-the-art ML models [18, 37, 38]. A limitation of these proposed attacks is the context in which they are built; their explicit goal is breaking classification tasks, rather than authentication tasks. There are two key differences:

1. Information returned from an authentication system is limited. In authentication tasks, class-level information is limited, and feedback from the model is essentially non-existent apart from the final authentication decision. During authentication, a user may attest by providing a credential-signature pair, but the authentication method's response is always either *YES* or *NO*. This is a much harder and more realistic case than assuming the system will return annotated labels–artifacts that are typically available in classification tasks [38].

2. Target information is secret and hidden. In prior ML attacks against binary classification models, an adversary may perturb the correctly-classified features of a known victim, and force it to be mislabeled [13], or generate new samples based on an already known set of victim samples [34]. However, in systems that use physical properties of the target for authentication, the adversary will not have access to the victim's signatures. Hence, the adversary is restricted to work with their own known signatures, which are incorrectly classified by an MDA system, when coupled with the victim's credentials. Very recently, this task has seen notable success in the biometric authentication setting, where model-based biometric authentication was fooled using random inputs [49].

In this paper, we examine and empirically test three MDA systems enabled by device fingerprinting schemes, which were published in the academic security literature. We develop an *untargeted exploratory attack* against these MDA systems, bootstrapping a simple sample crafting process through techniques inspired by program fuzzing. In addition, we adapt a complex zeroth-order optimization (ZOO) attack by Chen et al. [6], denoted Zoo, to the device authentication context. Interestingly, we find that device authentication can be easily fooled regardless of the attack complexity. We later use explainable AI (XAI) feature attribution techniques (i.e., approximation of the decision boundaries) and Local Intrinsic Dimensionality (LID) to investigate causes related to the adversarial subspace of each model [29, 41]. Our formal framework, coupled with the postmortem analyses, enables new design considerations that target three main areas of weakness in MDA systems: the signature generation algorithm (*SigGen*), the parameterization function (*Train*), and authentication heuristics (*Auth*). We adapt the randomized smoothing technique proposed by Cohen et al. [10] as a potential mitigation for neural network-based MDA systems, and find that it offers an encouraging direction for preventing masquerading attacks against potential MDA systems.

Our specific contributions follow:

(1) We develop the first empirical evaluation of device authentication methods in the context of semantically accurate, restricted query AML-style attacks and present a formal setting for them.
(2) We develop a comprehensive feature attribution analysis of successful adversarial samples from each attack, by looking at the interaction between subsets of device features and their influence on attack success. Particularly, we find that at most five features are needed for successful adversaries, even in the random fuzzing attack.
(3) We show that previous intuitions and assumptions in the model-based device fingerprinting literature do not align with the capabilities, goals, and postmortem artifacts of a restrictive query AML adversary. Notably, an authenticator's false positive rate can be degraded to between 24% and 67%, depending on the fingerprinting domain and query budget.
(4) Given the three proposed areas of weakness, we improve an existing MDA system by modifying two of them: *Train* and *Auth*. Likewise, we empirically illustrate that such an approach is viable for improving the false positive rate of the MDA system to just 1%.

2 BACKGROUND

We provide a brief overview of the enabling techniques behind device authentication, namely device fingerprinting, and adversarial machine learning.

2.1 Device Fingerprinting

Device authentication can be considered a general term for attesting a computational device's identity against some known signature. Device authentication is primarily enabled by device fingerprinting techniques. The exact method of extracting a device's fingerprint can vary, but the common goal is to create unique, attestable signatures that can be stored in a database and later queried for comparison. The most direct method of creating a device fingerprint is to leverage the *physical* manufacturing imperfections exhibited by hardware (i.e., process variation) to distinguish between devices. Bates et al. used USB enumerations between the USB controller hardware and the device operating system to accurately distinguish between devices of same make and model [4]. For wireless devices, techniques have leveraged process variation to accurately identify wireless sensor nodes [12], commodity devices [25, 28, 40], and radio network transmitters [32].

In terms of applications, device fingerprinting has become an enabling technology in the secure Internet of Things (IoT) literature. Although useful for device type identification and network-level access control [31], it has also enabled services such as quarantining [33] and watermarking [17]. Unfortunately, these applications are undermined if an adversary can reliably fool the underlying fingerprinting technique.

2.2 Adversarial Machine Learning

Most attacks against device fingerprinting tasks have been framed in terms of traditional, network-based adversaries, as categorized by Mamdouh, Elrukhsi, and Khattab [30]. Adversarial machine learning has shown that models such as artificial neural networks [46] and tree-based classifiers [27] are vulnerable to straightforward attacks.

For the authentication task, the adversary's goal is similar to mounting an attack within a limited query model, a setting that has been investigated by Dang et al. under the binary classification task [13]. However, attacks against authentication are particularly challenging given the lack of labeled information returned; this setting is analogous to the restricted query model where no counterexample is provided, as discussed by Angluin [2]. Several prior works have investigated techniques for fooling *biometric* authentication using the top-k scores from the system [43], or relying on complete knowledge about the victim [34]. Although similar, these works have not considered the *restricted-query binary decision* setting. This setting is much more realistic, as the adversary has no knowledge about the victim, and must interact with the authentication system to know if they are successful. A tangential result in fooling authentication systems is the work by Hitaj et al. [23], which uses generative models to guess string-based passwords. Contrary to their work, our method targets the signatures emitted by hardware, and leverages design faults of model-based fingerprinting techniques, rather than the low entropy of human-generated passwords.

The most similar work to ours is by Zhao et al. [49], who attack biometric authentication systems in the binary decision setting using randomly generated inputs. In contrast to their approach, our analysis is performed in the device authentication setting, using a plausible fuzzing-based attack and a gradient estimation attack.

For comparison, we discuss the result of random inputs on MDA systems and visualize the drawbacks of such an approach in Section 6.1.2 and Section 7. In addition, our analysis is grounded in approximations of the model's decision boundary (XAI) and the model's adversarial subspace (LID). Notably, LID helps us to quantify the acceptance region effect defined by Zhao et al. [49].

3 SECURITY MODEL OF AN MDA SYSTEM

In a non-adversarial setting, an MDA system involves several principals: an authentication server and multiple (honest) client devices. An MDA system operates in two phases.

(1) **Device registration.** Each device d is assumed to possess a unique device identification string id_d. Also, there exists some pre-defined mechanism that an authentication server can use to collect a set Y of data from device d and generate a signature sig_d, a string encoding of Y.
During *device registration*, the authentication server uses its device interface to collect a set T of $(\mathrm{id}, \mathrm{sig})$ pairs for all devices. We refer to the sig_d that is used during device registration as the representative signature of the device d. The set T constitutes the training data for the underlying ML model of the MDA system. The registration phase terminates when the training (testing and validation) of the ML model stops.

(2) **Device authentication.** We assume that only devices that are registered with the authentication server can query it for authentication. In the *device authentication* phase, the server uses the pre-defined mechanism to collect a (potentially different) set of data from the querying device d, and generates a new signature sig^*_d. The trained ML model processes the $(\mathrm{id}_d, \mathrm{sig}^*_d)$ pair to output a prediction vector. Given the prediction vector and the $(\mathrm{id}_d, \mathrm{sig}^*_d)$ pair as inputs, the MDA system uses some deterministic checks against certain threshold values to accept/reject the device.

Let us explain the working of an MDA system in a wireless network setting. Here, the network hosts (e.g., computers) act as clients to be authenticated, which can communicate directly with a dedicated authentication server. The clients wish to use resources on the network. As a preliminary challenge step, a host sends traffic over the wireless network to the dedicated server. The server uses this traffic to detect intrinsic timing data. It can process this timing data to generate a succinct fingerprint of the wireless network host. Radhakrishnan et al. show that this method is feasible using inter-arrival times of network packets, which enable identification and authentication of network hosts [40]. During device registration, an authentication server collects representative (host id, timing data) pairs of *each* host. It uses the collected data to train an underlying ML model. When a registered network device comes online again after some time, it generates a new set of traffic as part of the challenge, which emits a new signature. The trained ML model processes the new timing signature and outputs a prediction. The MDA system processes the prediction, along with the claimed host id, using a hand-crafted heuristic. It returns *YES* if it is a match, and *NO* otherwise.

3.1 Formalizing an MDA system

Fix integer t denoting the maximum number of times that any device interacts with the authentication server. Fix sets $\mathcal{I}$ and $\mathcal{S}$ to denote the set of all (string encodings of) valid device identifiers, and the set of all (string encodings of) valid signatures, respectively, for any given MDA system. Let $n = |\mathcal{I}|$ denote the number of elements in set $\mathcal{I}$. Fix $\mathcal{C}$ as the set of functions that encode the mapping $\mathcal{I} \times \mathcal{S} \rightarrow \mathbb{R}^n$. We define an MDA system AS over sets $(\mathcal{I}, \mathcal{S}, \mathcal{C})$ as a tuple of algorithms (*SigGen*, *Train*, *Auth*) with the following syntax.

SigGen: The randomized *signature generation algorithm SigGen* : $\mathcal{I} \rightarrow \mathcal{S}^t$ takes a device-identification string $id \in \mathcal{I}$ and returns a list $L = (\mathrm{sig}_1, \mathrm{sig}_2, \ldots, \mathrm{sig}_t)$ of device signatures, where $\forall i \in [1, t], \mathrm{sig}_i \in \mathcal{S}$. We assume that $L[1] = \mathrm{sig}_1$ is the representative signature of the device and is used by AS for registration; when $j > 1$, $L[j]$ is the signature that will be used by AS to authenticate the device in its j-th attempt.

Train: The randomized *model training algorithm Train* : $(\mathcal{I} \times \mathcal{S})^n \rightarrow \mathcal{C}$ takes a tuple of $(\mathrm{id}, \mathrm{sig})$ pairs of training data and returns a trained model $C \in \mathcal{C}$. The model C takes $(\mathrm{id}, \mathrm{sig})$ as inputs and returns a prediction vector P, where each element $P_{\hat{\mathrm{id}}}$ in P denotes the prediction/confidence value that $\mathrm{id} = \hat{\mathrm{id}}$. Since an MDA system registers *all* devices in the registration phase, we insist *Train* to take n tuples as input.

Auth: The deterministic *device authentication algorithm Auth* : $\mathcal{I} \times \mathcal{S} \times \mathcal{C} \rightarrow \{0, 1\}$ takes a $(\mathrm{id}, \mathrm{sig})$ pair and a trained model C as inputs and returns a binary decision: 0 indicating "fail" and 1 indicating "pass." The authentication server uses *Auth* to authenticate any device $(\mathrm{id}, \mathrm{sig})$ using C and certain decision heuristics (that are part of the description of *Auth*). Note that C will always be a fixed input to the *Auth* algorithm.

ML models seldom have 100% accuracy. Therefore, we define δ-correctness of an MDA system (as opposed to absolute correctness). In words, a δ-correct AS is one whose *Auth* algorithm outputs one for valid $(\mathrm{id}, \mathrm{sig})$ pair with probability at least $(1 - \delta)$. Natural examples[1] of AS are $(\mathrm{id}, \mathrm{sig})$ pairs (where $\mathrm{sig} \notin SigGen(\mathrm{id})$) for which *Auth* outputs one with probability at least $(1 - \delta)$. In practice, δ-correctness can be described using the standard definitions of *accuracy, precision, and recall.*

3.2 Threat model

In the adversarial setting, an adversary knows the underlying signature-generation algorithm of an MDA system AS and gets *only* decision-level access to $Auth(\cdot, \cdot, C)$, where C denotes a trained model. It is given *no* other information. Such an adversary is referred to as an untargeted exploratory (UE) adversary by Biggio et al. [5]. Informally, the security goal of AS is to prevent UE adversaries A from *fooling AS* by making $Auth(\cdot, \cdot, C)$ output one for some id of an honest user, and some arbitrary signature that the adversary generates (from its own signature).

A standard assumption is that an adversary owns one of the devices that is registered with the (trusted) authentication server. Let id_A be the device identifier and $L_A = SigGen(\mathrm{id}_A)$ be the list of signatures of the device that A owns. Given only decision-level

[1] Hendrycks et al. [22] refer to such inputs as natural adversarial examples.

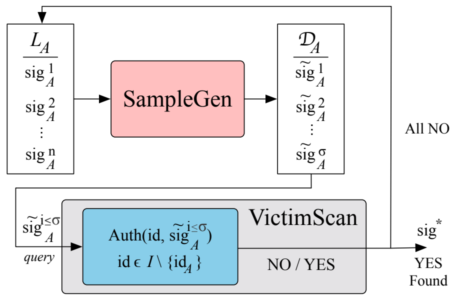

Figure 1: Visual description of the attack framework. Adversary A uses *SampleGen* to generate (multiple) adversarial samples $\widetilde{\mathrm{sig}}_A^{i\le\sigma} \in \mathcal{S}$ and store them in a set $\mathcal{D}_A$. It uses *VictimScan* to scan and select a victim device (id), and then pairs the device with a signature (sig^*) from $\mathcal{D}_A$ to fool *Auth*.

access to *Auth*, we say A breaks the security of *AS* or *fools AS* if and only if A outputs an $(\hat{\mathrm{id}}, \hat{\mathrm{sig}})$ pair and $Auth(\hat{\mathrm{id}}, \hat{\mathrm{sig}})$ returns one, where $\hat{\mathrm{id}} \in \mathcal{I} \setminus \{\mathrm{id}_A\}$ and $\hat{\mathrm{sig}} \in \mathcal{S}$. In terms of the vocabulary from recent work [43], when A fools *AS*, it is said to perform a successful *dodging* attack against *AS*. The robustness of an *AS* against an UE adversary A is defined as the probability with which A fails to fool *AS*. The failure rate of *AS* can be succinctly described in terms of the false acceptance rate, or *false positive rate*.

4 FOOLING MDA SYSTEMS

Our central hypothesis is as follows: *MDA systems can be reliably fooled due to the mis-characterization of adversarial capabilities.* From a design standpoint, MDA systems should have built-in robustness against UE adversaries which can provide perturbed samples to the underlying fingerprinting technique. As noted previously in Section 2, recent work in device fingerprinting has not considered an adversarial machine learning style of attack, which can iteratively modify a sample based on the system's feedback.[2] We describe a framework for such attacks in the following text.

4.1 Attack Framework

Let id_A denote the identifier of the device that the adversary controls. Let $L_A = SigGen(\mathrm{id}_A)$, and *Perturb* be some randomized procedure for generating a perturbed signature from an input signature. At a very high level, untargeted-exploratory attacks comprise two algorithms: *SampleGen* and *VictimScan*. The *SampleGen* algorithm takes as input the list L_A of device signatures and returns a set of adversarial samples $\mathcal{D}_A := \{\widetilde{\mathrm{sig}}_A^i : i \in [0, \sigma], \widetilde{\mathrm{sig}}_A^i \in \mathcal{S}\}$. The *VictimScan* algorithm is responsible for scanning the set of known device identifiers to find an $(\mathrm{id}_v, \mathrm{sig}^*)$ pair that fools *AS*. The behavior of *VictimScan* can vary based on the implementation, as it determines how potential victims are sampled from the known set of device identifiers. The hyperparameter σ acts as an upper bound on the number of attempts. Here, sig^* is sampled from $\mathcal{D}_A$; id_v is sampled from $\mathcal{I} \setminus \{\mathrm{id}_A\}$. Both algorithms get decision-level access

[2]Due to the need for feedback, our attack framework does not directly translate to device fingerprinting systems deployed only for reconnaissance or risk assessment purposes. The analysis of these settings is an interesting direction for future work.

Algorithm 1 QuickFuzz, our fuzzing-based implementation of *SampleGen* for adversarial sample-crafting.
Input: L_A, a set of initial samples owned by A. σ, an arbitrary upper bound on number of adversarial samples to create, and algorithm *Auth*, an interface available for querying *AS*.
Output: Set of adversarial samples, $\mathcal{D}_A$.

$\mathcal{D}_A \leftarrow \varnothing$
$R \leftarrow$ YES
$v \leftarrow 0$ ▷ (Distortion parameter)
while $|\mathcal{D}_A| < \sigma$ **do**
  sig $\leftarrow$ *random_sample*(L_A)
  ▷ (Loop until we meet sufficient distortion for a *NO* result.)
  while $R =$ YES **do**
    increment(v)
    $\widetilde{\mathrm{sig}}_A^i \leftarrow \mathrm{Perturb}(\mathrm{sig}, v)$
    $R \leftarrow Auth(\mathrm{id}_A, \widetilde{\mathrm{sig}}_A^i)$
  end while
  append($\mathcal{D}_A, \widetilde{\mathrm{sig}}_A^i$)
end while
return $\mathcal{D}_A$

to *Auth*. See Figure 1 for a visual description of the described attack framework.

In this work, we explore two instances of *SampleGen*: a new algorithm, QuickFuzz and Zeroth-Order Optimization (Zoo) [6]. We keep *VictimScan* fixed for simplicity, and describe it in the context of attack scenarios in Section 5.2. Our selection is motivated by the need for two distinct sample generators that can empirically characterize A within our framework.

4.1.1 *QuickFuzz.*

We describe QuickFuzz in detail in Algorithm 1. The intuition behind the design of QuickFuzz is to perform a discrete, uniform random walk through the signature space, using the provided input signature as a starting point. This approach is inspired by program-fuzzing attacks in the systems literature. In the outer loop of Algorithm 1, *SampleGen* generates several adversarial samples using the adversary's own list of signatures as a seed; in the inner loop, it invokes a generic *Perturb* subroutine that can be tuned for specific fingerprinting domains. The effective goal of the inner loop is to reach a *NO* result, which signals that the adversary has crossed their own decision boundary, and potentially entered another user's decision space. Thus QuickFuzz assumes the MDA system acts as a one-vs-rest classifier internally. In our implementation, *Perturb* offsets a small, random subset of attributes in sig, each by an amount δ that follows a uniform distribution, i.e., $\delta \sim \mathcal{U}(a, b)$ for lower bound a and upper bound b. In practice, b is an evaluation of $f(v)$, an upper bound function controlled by parameter v in Algorithm 1. In each iteration of the inner loop, the upper bound increases by a function of v, thus, acting as a search step-size parameter. The step-size is a trade-off between granularity of the search through signature space and number of queries made to *Auth*. For any attribute value $x_i \in \mathrm{sig}_A$ of the associated pair $(\mathrm{id}_A, \mathrm{sig}_A)$, the perturbed attribute value $x_i + \delta$ is clipped so that the resulting pair $(\mathrm{id}_A, \widetilde{\mathrm{sig}}_A)$ is still in the domain of C.

In practice, a and $f(v)$ are initiated by experimenting with the adversary's own signatures, since the adversary simply notes how many queries it takes to achieve a *NO* result. The function $f(v)$ can be defined separately for each fingerprinting domain, as each may rely on different ranges of feature attribute values. These are provided in subsequent sections.

4.1.2 Zoo. Given only query access to a function, Zeroth Order Optimization (ZOO) is a method for performing optimization of that function [35]. Recently, Chen et al. [7] demonstrated attacks on ML models using ZOO. Unlike transfer-based attacks which approximate the victim model with a separate substitute model, a ZOO attack approximates the victim model's gradient directions through a series of specially-crafted queries. Most zeroth-order attacks can leverage probabilities returned from the model, which are known as score-level attacks. HopSkipJumpAttack [6], which is the variant we use in our experiments, only requires the top-1 label prediction from the model to be successful. This variant of ZOO attack is known as a decision-level attack, since it only requires the model decision to approximate the best gradient direction. Thus, it is compatible with our hard-label authentication threat model. Specifically, we leverage the untargeted version presented by Chen et al. to act as a walk through the signature space, guided by the approximated gradient information. We denote the HopSkipJumpAttack in our discussion as Zoo. We use the Cleverhans adversarial machine learning library [36] as the reference implementation of HopSkipJumpAttack. This implementation was originally for image classification adversaries; we modify it to operate over the domain of *Auth* instead.

In our experiments, for simplicity, *VictimScan* will select the victim devices in a deterministic fashion. With the default Cleverhans hyperparameters, Zoo requires several thousand queries to find *YES* samples. Likewise, it is necessary to empirically tune the hyperparameters over many queries. Due to the complexity of Zoo and different authentication system characteristics, it took between 500 to 3000 queries to initialize hyperparameters that made Zoo competitive. This is a *downside* of using gradient estimation techniques for the authentication setting. We do not count these queries in later tallies in order to make the comparison with QuickFuzz consistent.

4.2 Metrics

We define two metrics to help us explain the effectiveness of the QuickFuzz and Zoo attacks against an MDA system that is defined over sets $(\mathcal{I}, \mathcal{S}, \mathcal{C})$.

4.2.1 False-positive rate. First let us define a *run* as a process where each user in the system attempts to attack every other user, one user at a time. We define the false-positive rate α of a run as $\alpha = \frac{r}{n^2-n}$, where $r \leq (n^2 - n)$, denotes the number of successful attacks across all attacks in the run. Intuitively, this is equivalent to taking the average of an $n \times n$ adversary-victim Boolean matrix (1 denotes successful attack, and 0 for failure), but ignoring the diagonal values. In our experiments, we average α over three runs. In each run, we swap a (fresh) victim's device with the adversary's device and run the attack. This effectively gives us the average efficiency of an attack.

4.2.2 Distortion. Fix some $\text{sig}_j \in \mathcal{S}$, and some randomized *Perturb* procedure. Let $\text{sig}_j^* = Perturb(\text{sig}_j)$. We define the distortion ϵ_j of the j-th sample pair $(\text{sig}_j, \text{sig}_j^*)$ as $\epsilon_j = (||\text{sig}_j - \text{sig}_j^*||_2)/||\text{sig}_j||_2$. We also define the average distortion $\bar{\epsilon}$ across multiple (say, p) runs of *Perturb* as $\bar{\epsilon} = (\sum_{i=1}^{p} \epsilon_j)/p$.

5 SETUP

Our experiments are motivated by the following two research questions related to the central hypothesis:

(1) What properties of an MDA system *AS* are "exploitable" by an adversary in our attack framework? In particular, do *SigGen*, *Train* and *Auth* — algorithms that define *AS* — expose attack surfaces that an adversary can access?
(2) To what extent are these properties exploitable? For example, how many queries suffice to fool *AS*?

We answer these two questions by attacking three systems: USB-Fingerprinting [4], GTID [40], and WDTF [11]. Each of these three systems provide fingerprinting facilities that can form the foundation of an MDA system. Notably, every system has been recently published at academic security venues. We selected these three systems as: (a) each tackles a significantly different fingerprinting domain, (b) the description of each system was clearly specified, and (c) raw data was available for all three systems, in contrast to deployed authentication systems which use inaccessible models and data.

5.1 Target MDA systems

We explain each MDA system by describing its three component algorithms: *SigGen*, *Train* and *Auth*. Specific implementation details are available in Section A.1 of the Appendix.

5.1.1 USB-Fingerprinting [4]. In this MDA system (denoted USB-F), *SigGen* uses specific USB-enumeration timings of a computer under test to generate device signatures. The *Train* algorithm trains the underlying ML model (Random Forest) in a *target* vs. *outlier* fashion. It creates a new model for every device that registers on the system, and balances outlier classes with respect to every possible device in the system. *Train* also performs over-sampling (with replacement) until the number of target samples matches the number of outlier samples. The *Auth* algorithm uses majority voting over multiple signatures to make a pass/fail decision.

5.1.2 GTID [40]. In this MDA system, *SigGen* uses inter-arrival times of network (TCP) packets to generate device signatures, for both device authentication and device-type authentication. The *Train* algorithm uses Artificial Neural Networks (ANNs) as the underlying ML model. It uses an ensemble approach by training one ANN for device identification, and another ANN for device-type identification. Although tested as a fingerprinting technique, GTID is advertised by the authors as a potential authentication system [40]. Thus, we defined *Auth* as follows. A device passes authentication only if the GTID heuristics (Algorithm 1 from Radhakrishnan et al. [40]) output a predicted device ID and device type that matches the user's claimed device id *and* device type.

5.1.3 WDTF [11]. In this MDA system, *SigGen* uses probe-request traffic of IEEE 802.11 wireless devices to generate device signatures.

System	$\delta \sim \mathcal{U}(a, f(v))$
GTID	$\delta \sim \mathcal{U}(0, 2 \times (v+1))$
USB-F	$\delta \sim \mathcal{U}(10^{-25}, 10^{-20+v})$
WDTF	$\delta \sim \mathcal{U}(10^{-19}, 10^{-17+v})$

Table 1: Lower and upper bounds of the uniform distribution used by QuickFuzz's *Perturb* implementation, as described in Section 4.1.1.

The authors use a customized statistical model based on some distance function κ (that is parameterized using the representative device signature of all devices) to instantiate *Train*. The *Auth* algorithm uses κ to compute a list of distance-values between the signature of the device under test and the representative signature of all registered devices. If none of the distance values are less than some pre-determined threshold value, then *Auth* returns 0; otherwise, it returns 1.

We provide the function $f(v)$ used in each QuickFuzz attack of the three systems in Table 1. These values were found empirically with knowledge of *Sup*(*SigGen*), using between 5 to 20 queries to find a reasonable step-size before achieving *NO* from *Auth*. These queries do not have to be performed by the adversary in a single session, so they are not included in later query tallies.

5.2 Attack Scenarios

Each of the described systems are evaluated under different attack scenarios — described using *SampleGen* and *VictimScan* — to answer our core research questions. Each scenario uses the metrics from Section 4.2. Note that *SampleGen* gets $L_A = SigGen(\mathrm{id}_A)$ as input and returns the adversarial-sample set $\mathcal{D}_A$. Regardless of the attack scenario, we always measure against the same unseen test set to give a fair comparison. In addition to our metrics, we rely on the notion of accuracy and recall defined by the scikit-learn library,[3] and calculate them in attack scenarios by including any found adversarial samples (with replacement) into the test set during evaluation.

5.2.1 Baseline. In the baseline case, *SampleGen* initializes $\mathcal{D}_A$ with *all* signatures sig_A in the list L_A. Likewise, *Perturb* is simply an identity function. The *VictimScan* fixes an arbitrary set $\mathcal{V} \subseteq \mathcal{I}$ of victim devices, and queries *Auth* by sampling (with replacement) id from $\mathcal{V}$ and sig from $\mathcal{D}_A$. In our evaluations, the baseline case will be used to establish the lower bound on the system's robustness.

5.2.2 Random. Let $\min(L_A), \max(L_A) \in \mathbb{R}$ denote the minimum and maximum signature values in L_A. Let m be some positive integer. In the random test case, *SampleGen* samples m signatures uniformly at random from the range $\min(L_A), \max(L_A)$ and stores them in set $\mathcal{D}_A$. The *VictimScan* algorithm is same as that of the baseline case. Although random samples may fool the underlying model, they can be easily detected by the MDA system using adversarial training [46]. As we show in later sections, the random strategy is also not viable for maximizing α.

[3]https://scikit-learn.org/stable/modules/model_evaluation.html

5.2.3 Greedy and exploratory. In both the greedy and exploratory cases: the *SampleGen* algorithm can be instantiated with either QuickFuzz or Zoo; the *VictimScan* algorithm is the same as that of the baseline case. The difference between greedy and exploratory cases is that in the latter, the number of queries that the adversary can make to *Auth* is bounded by a pre-determined threshold value that is less than the total number N of possible queries. ($N = |\mathcal{D}| \times |\mathcal{I}|$). On the other hand, in the greedy case, the adversary is allowed to make N queries to *Auth*. Moreover, in the exploratory case, we are interested in computing the number of victim devices that the adversary can masquerade. Hence, the adversary tries other victim devices until it exhausts its query budget. That means, even when its first query to *Auth* is successful, it continues attempting to masquerade other victim devices. In the greedy case, the adversary stops as soon as *Auth* returns one.

5.3 Attack post-mortem

In this section, we discuss three metrics that will be useful to carry out a post-mortem analysis of any attack that is captured by our attack framework. We allow the post-mortem analyst to have full access to all three algorithms — *SigGen*, *Train*, and *Auth* — that define an MDA system.

5.3.1 Local Intrinsic Dimensionality — exploiting expected dimensionality of SigGen. The authors of each fingerprinting system apply expert knowledge of their respective domains to design classifiers that can overcome the variance of emitted signatures. However, they do not consider the curse of dimensionality, which states that as dimensionality of a data set increases, the effectiveness of distance-based measurements (generally) decreases. The intrinsic dimensionality (ID) of a data set quantifies the relationship between the dimensionality of a data set and the effect on distance-based measurements [24]. This notion extends to the local continuous intrinsic dimension (LCID) around a point in the set using nearest-neighbor distances. An approximation of the LCID value is known as the Local Intrinsic Dimensionality (LID). Recently, Amsaleg et al. showed that LID values can be used to explain successful adversarial attacks against ML models [1]. More precisely, they showed that as LID increases, the amount of perturbation needed to move into an adversarial region decreases. In our post-mortem analysis, we study the unperturbed signature samples and the adversarial samples that are generated using different instances (QuickFuzz and Zoo) of *SigGen*. At a high level, LID quantifies the adversarial subspace within each model. This is analogous to the acceptance region effect showcased by Zhao et al. in biometric authentication systems [49].

5.3.2 Feature Attribution — exploiting attack surface of Train. Given the heterogeneity of the underlying ML models that the MDA systems use, we must abstract away much of the model-specific behavior already covered in the literature [18, 27, 46]. Although the MDA systems seem incomparable on the surface, they share the common assumption that devices are unique due to hardware-manufacturing imperfections. These imperfections are thought to be captured by the signature-generation algorithm, *SigGen*.

Our analysis of features of the (id, sig) pairs is enabled by recent work in the Explainable AI (XAI) literature [15, 20, 41]. The main advantage of leveraging XAI is homogenizing the discussion between heterogeneous methods and data, by *not* biasing analysis towards any particular model architecture's "naive" explainability (i.e., decision trees and hand-crafted heuristics). This motivates our use of Local Interpretable Model-agnostic Explanations (LIME), an XAI technique proposed by Ribeiro, Singh, and Guestrin [41]. LIME is a good match for our setting as: (a) it is model agnostic (only requires score-level access to the trained model) and (b) it can generate explanations on a per-adversarial sample basis.

Given an ML model, LIME generates a linear approximation of the model and allows computation of attribution scores for each feature in the original model. In the MDA setting, we use the adversarial signatures – generated by QuickFuzz and Zoo – as inputs to LIME to get a linear approximation of the trained MDA system's model (i.e., the output of *Train*). Based on the attribution scores that LIME outputs, we analyze the features that make an MDA system susceptible to AML-style attacks. It is worth noting that recent works have attempted to subvert the explanation ability of XAI methods, namely back-propagation-based interpreters [14, 48] and LIME [44]. In our setting, the post-mortem analysis uses LIME honestly.

6 RESULTS

We organize our discussion using the research questions posed in Section 5 as a road map.

6.1 Attack Effectiveness

We examine the effect of *SampleGen* instantiations combined with the defined strategies.

6.1.1 Performance Impact. In Tables 2, 3, and 4, we compare against the published results, our baseline, and attack scenarios for USB-F, GTID, and WDTF, respectively. For USB-F in Table 2, we find that our implementation exceeds the published results in the regular device identification scenario described by Bates et al. [4]. When under the Random attack described in Section 5.2.2, the system admits no adversaries (89% accuracy, 0% recall). However, the accuracy and recall diminish reliably as any given instance of A expands their search. A greedy adversary manages to reduce accuracy down to 80% using QuickFuzz, lower than the 93% with Zoo. However, as the query budget is relaxed, Zoo ultimately overtakes QuickFuzz in terms of accuracy. In the worst case, the 500-query adversary with Zoo will reduce accuracy to 59% and recall to 52%. This contrasts with the case of GTID in Table 3, as QuickFuzz is generally more successful regardless of query budget. We note that our implementation of GTID had very close performance to the published result of 99% accuracy, with only some false negatives (83% recall versus published 94% recall). We observed that during successful attacks, the device id ANN was sufficiently confident in adversarial samples to bypass the UNKNOWN path of Algorithm 1 from Radhakrishnan et al. [40]. On the surface, this appears to reflect similar vulnerabilities in multi-modal biometric authentication [26, 42]. Essentially, the adversary instance A can exploit failure modes of the *Auth* algorithm within the GTID MDA system. Interestingly, Zoo is more successful in all adversarial cases of WDTF in Table 4. In fact, accuracy for WDTF fell to 25%, accompanied by a recall score of 0% in the worst case. Even the greedy adversary case is successful against WDTF, with an accuracy of 69%. Despite the three MDA systems relying on different pattern recognition techniques, each was susceptible to the tested *SampleGen* instances.

	Accuracy	Recall
Bates et al. [4]	94-99%	-

	QuickFuzz		ZOO	
	Accuracy	Recall	Accuracy	Recall
Our Baseline	100%±0%	100%±0%	100%±0%	100%±0%
Random	89%±0%	0%±0%	89%±0%	0%±0%
Greedy A	80%±2%	0%±0%	93%±2%	56%±9%
Exp. $A, \bar{p} = 100$	80%±2%	0%±0%	92%±2%	52%±10%
Exp. $A, \bar{p} = 300$	70%±2%	0%±0%	63%±3%	52%±19%
Exp. $A, \bar{p} = 500$	68%±4%	0%±0%	59%±2%	52%±14%

Table 2: Performance comparison of USB-F scenarios. Measurements are averaged over three runs.

	Accuracy	Recall
Radhakrishnan et al. [40]	99%	94%

	QuickFuzz		ZOO	
	Accuracy	Recall	Accuracy	Recall
Our Baseline	98%±1%	83%±7%	98%±1%	83%±7%
Random	87%±0%	7%±0%	87%±0%	7%±0%
Greedy A	86%±0%	0%±0%	90%±1%	0%±0%
Exp. $A, \bar{p} = 100$	82%±1%	0%±0%	93%±0%	0%±0%
Exp. $A, \bar{p} = 300$	71%±1%	0%±0%	89%±0%	0%±0%
Exp. $A, \bar{p} = 500$	70%±2%	0%±0%	88%±0%	0%±0%

Table 3: Performance comparison of GTID scenarios. Measurements are averaged over three runs.

	QuickFuzz		ZOO	
	Accuracy	Recall	Accuracy	Recall
Our Baseline	100%±0%	100%±0%	100%±0%	100%±0%
Random	77%±3%	83%±12%	77%±3%	83%±12%
Greedy A	69%±0%	0%±0%	56%±0%	0%±0%
Exp. $A, \bar{p} = 100$	67%±6%	0%±0%	56%±0%	0%±0%
Exp. $A, \bar{p} = 300$	62%±0%	0%±0%	29%±6%	0%±0%
Exp. $A, \bar{p} = 500$	62%±0%	0%±0%	25%±0%	0%±0%

Table 4: Performance comparison of WDTF scenarios. Measurements are averaged over three runs.

6.1.2 Attack Distribution & Queries. We delve further by examining the spread of damage to the MDA systems when under attack. This is visualized for each system in Figures 2, 3, and 4. The access matrices allow us to make high level observations about the *SampleGen* instances. The top and bottom rows of the access matrices correspond to the QuickFuzz and Zoo *SampleGen* instances, respectively. Each access matrix represents the average over three random seeds of running an attack scenario, where each adversary

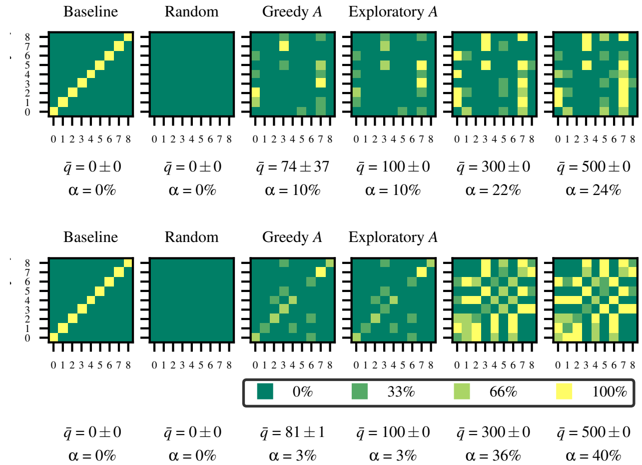

Figure 2: Access matrices for USB-F users, averaged over three random seeds for QuickFuzz (top) and Zoo (bottom).

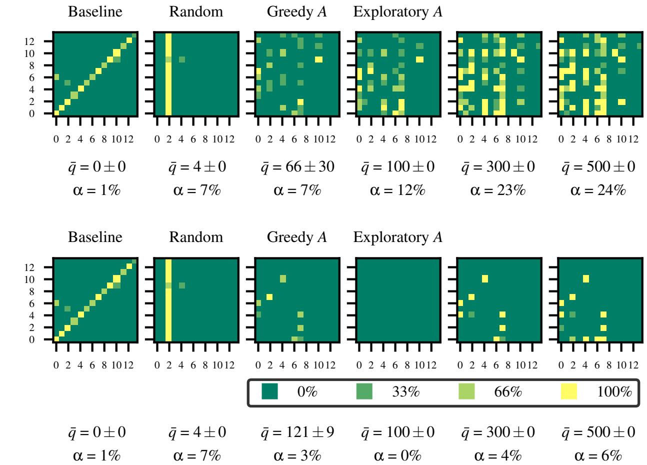

Figure 3: Access matrices for GTID users, averaged over three random seeds for QuickFuzz (top) and Zoo (bottom).

instance with identifier id_A occupies a row, and each column represents attempts on victim id_v. Color intensity corresponds to success rate.

Notably, some victims tend to be more susceptible than others, and likewise, some adversaries tend to be stronger than others. This behavior differs depending on the exact attack used. For instance, QuickFuzz on USB-F in Figure 2 allowed increasingly higher access to victims zero, three, and seven as queries were increased. This behavior amplifies with Zoo, allowing increased access to additional victims two, four, and six. False positive rate δ is notably higher using Zoo, peaking at 40% in the worst case, despite requiring more queries than QuickFuzz. A similar pattern can be observed with the random attack on GTID, shown in Figure 3. Victim two is more susceptible to the random attack than any other principal. Unlike with USB-F, QuickFuzz was generally more successful against GTID, even requiring less queries, and spreading the attack impact more evenly. Even with 100 queries, the QuickFuzz attack manages to fool GTID with a δ of 12%. Zoo manages to outperform QuickFuzz when used on WDTF, as shown in Figure 4. WDTF is the most susceptible system, reaching a δ of 67% in the worst case. Overall, tests of the two attacks against the three systems show that a trade-off exists between number of queries and success. The more complex Zoo attack is not necessarily more successful in every case, as it sometimes requires more queries or fails to find failure modes of the MDA system.

6.2 Feature Attribution Analysis

As described in Section 5.3, we aggregate positive feature attribution weights across every victim in each attack scenario, to visualize the features that were most important in influencing a decision. For ease of comparison and visualization, we select up to eight victims and their perturbed samples from the Exploratory A scenario ($\bar{p} = 300$), and query LIME to return only the top-eight features attributing to decisions.

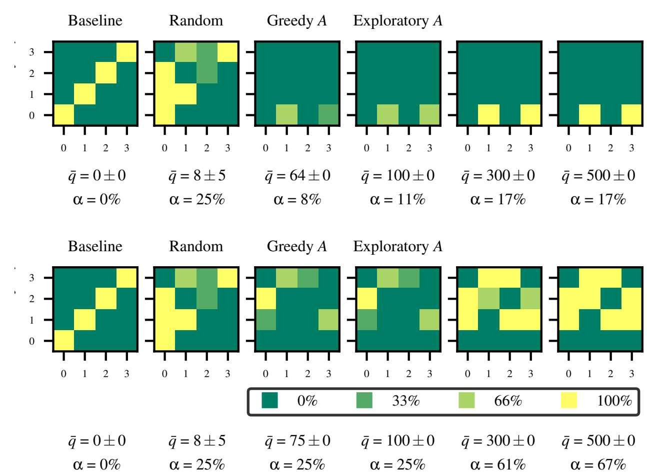

Figure 4: Access matrices for WDTF users, averaged over three random seeds for QuickFuzz (top) and Zoo (bottom).

6.2.1 USB-F. Figure 5 shows feature attribution weights for up to eight victims under the USB-F method with QuickFuzz on top, and Zoo on the bottom. We see that the attribution weights returned by LIME vary based on the attack. Samples generated with QuickFuzz offered lower weights compared Zoo. In the case of Zoo, we can make some high level observation. Mainly, any arbitrary adversarial sample only relied on a handful of features, according to LIME. For example, victim 'vatta' was subverted with feature subset $\{x_{12}, x_{18}, x_{23}, x_{24}\}$. In fact, this pattern is not unique to 'vatta.' Adversarial samples needed no more than five of the features to align with the victim to be successful. Of interest is that features

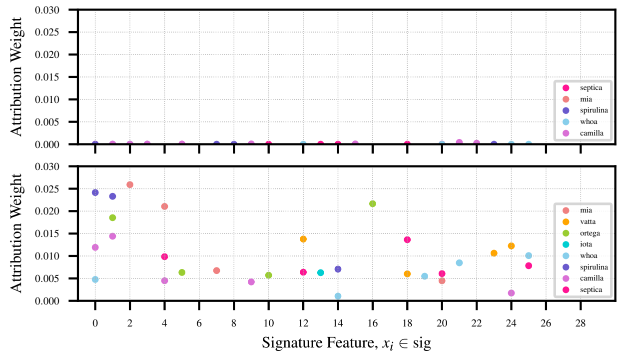

Figure 5: USB-F feature attribution weights for victims across 29 features with QuickFuzz (top) and Zoo (bottom).

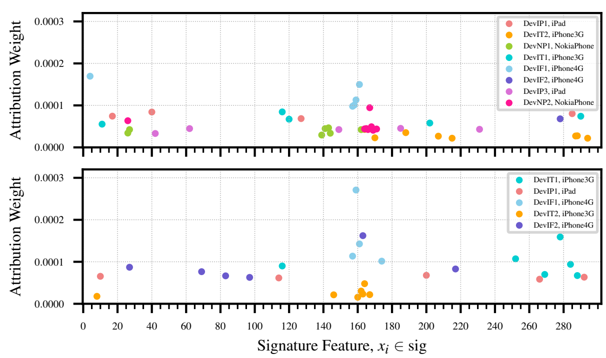

Figure 6: GTID feature attribution weights for victims across 300 features with QuickFuzz (top) and Zoo (bottom).

$\{x_{26}, x_{27}, x_{28}\}$ were never used. Since attacked features vary between victims, it is not clear that domain knowledge would help secure the model.

6.2.2 GTID. The feature attribution for GTID across victims is shown in Figure 6. In either case, we notice that attributions across victims is clustered relatively close to the center. Since each attack had similar scaling of attribution weights, we can now see that specific features vary between attacks. For example with Zoo, 'DevIT2, iPhone3G' was subverted with features in the center of the signature, while features in the periphery were used with QuickFuzz. This behavior is seen with other victims, notably 'DevIT1, iPhone3G' and 'DevIF2, iPhone4G'. Some overlap occurs with 'DevIF1, iPhone4G' as features were selected towards the center. Radhakrishnan et al. select the start and end points of the signature histogram to fit the peak of tested traffic (see Footnote 3 [40]). It is possible that the model attends disproportionately to the central peak of a typical signature, rather than the periphery. Apart from this spatial correlation between original sample and feature attributions, there is little relation between the perturbed features and the packet arrival-time semantics. Likewise, only a handful of features are chosen in either attack, as shown prior with USB-F. We

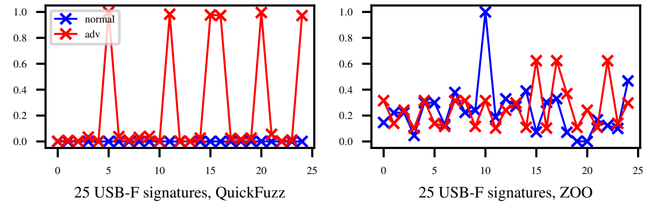

Figure 7: Approximation of LID over 25 randomly chosen samples from USB-F for QuickFuzz and Zoo.

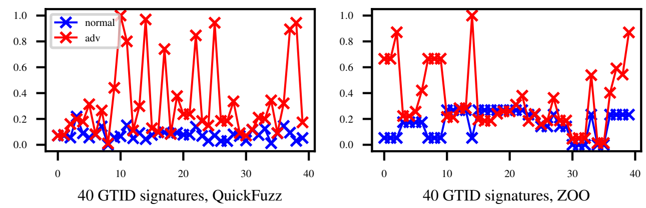

Figure 8: Approximation of LID over 40 randomly chosen samples from GTID using QuickFuzz and Zoo .

conclude that the detection of these brittle features during design would be non-obvious.

6.2.3 WDTF. Feature attribution weights for the four victims under WDTF were very similar to USB-F and GTID, and are omitted for brevity. We observed that the adversary only needs a handful of important features for the attack to succeed. This was true regardless of the considered victim. Likewise, features also varied between each attack, so connection back to domain semantics would not be helpful.

6.3 Local Intrinsic Dimensionality (LID)

In the previous section we saw that component-wise feature analysis is difficult, as each instance of *SampleGen* exploited different features. We abstract away features and instead focus on analyzing the adversarial subspace of inputs to each MDA, as described in Section 5.3. Figures 7 and 8 show the LID values of randomly selected normal (blue) and adversarial (red) *SampleGen* signatures from the USB-F and GTID MDA systems, respectively. For brevity, we omit results from WDTF, as they mirrored those of GTID. As in the analysis performed by Ma et al. [29], we use min-max normalization to obtain LID scores in the range $[0.0, 1.0]$.

Our results are consistent with the findings of Ma et al., which is that adversarial regions in the signature space can be characterized by equal, or higher, LID scores than normal data regions. This effect is primarily noticeable with QuickFuzz for both USB-F and GTID, and Zoo on GTID. We interpret this as follows. Although expert knowledge can arrive at intuitive features that are useful for classification, they contain a (local) intrinsic dimensionality that increases as points move away from the original *SigGen* output

	QuickFuzz	ZOO
USB-F	169%±529%	8.36%±7.68%
GTID	255%±474%	66.3%±81.1%
WDTF	2.30%±2.29%	93.8%±53.4%

Table 5: Average distortion ϵ induced by each SampleGen instance of QuickFuzz and Zoo.

space. It suffices to move in random directions to find local submanifolds with high complexity, as evidenced by the success of QuickFuzz. Zoo on USB-F had mixed results, with LID fluctuating between normal and adversarial levels, while Zoo on GTID behaves as expected. One possible explanation is that Zoo unintentionally crafts samples with lower LID during gradient approximation.

6.4 Attack Distortion Characteristics

We empirically evaluate the average distortion induced by each *SampleGen* instance of QuickFuzz and Zoo in Table 5. Generally, the QuickFuzz instances produced high values of $\bar{\epsilon}$ with high variability. Zoo tends to have lower distortion on USB-F and GTID with lower variation, but increases with WDTF. We interpret this as a by-product of the methods. WDTF exhibited high false-positives to random samples earlier in Section 6.1, whereas USB-F did not. We essentially have two extreme cases of robustness to random uniform noise, with USB-F responding positively, GTID performing moderately, and WDTF responding negatively. In this sense, a random walk should induce more noise with USB-F than the guided search of Zoo. WDTF only requires 2.30% distortion on average with random walk, but needs 93.8% with Zoo. This indicates our *SampleGen* algorithms cover two distinct strategies. The strategy to choose depends on the data *Train* was instantiated with.

7 MITIGATIONS

Our analysis of MDA adversaries in the previous section poses an interesting challenge. Although XAI can tell us which features are brittle, the exact features tend to vary between instances of *SampleGen*. Abstracting away features and focusing on the adversarial manifold enables the use of LID. Although encouraging, LID was more stable with Zoo than with QuickFuzz. Thus we attempt to protect an existing MDA system without relying on knowledge of the domain or assumptions of the underlying manifold. In this section, we rely on state-of-the-art results obtained by randomized smoothing [10]. Cohen et al. show that randomized smoothing can efficiently certify robustness given a fixed parameter σ which bounds the size of distortion an adversary induces.

We examine the effect of the two *SampleGen* instances when *Train* is modified according to the randomized smoothing technique. Notably, this defense is viewed as modifying *Train*'s output, due to training with Gaussian data augmentation. We apply the randomized smoothing technique to GTID to obtain the **smooth-GTID** MDA system. In practice, we remove the heuristics proposed by the original authors of GTID in favor of those presented by Cohen et al. due to effects noted in Section 6.1.1. In this instance, randomized smoothing necessitates modification of *Auth*, although it may not be necessary in all cases.

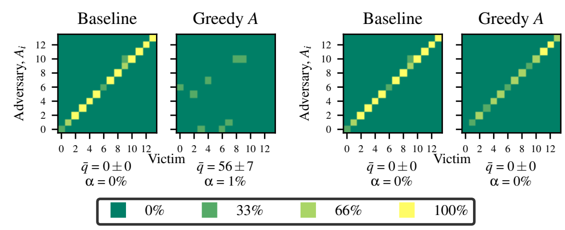

Figure 9: Access matrices for smooth-GTID over three random seeds with QuickFuzz (left) and ZOO (right), for $\sigma = 0.5$.

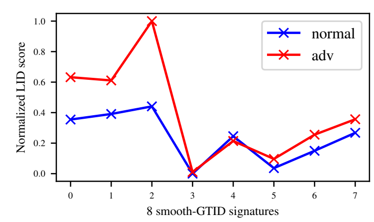

Figure 10: Approximation of Local Intrinsic Dimensionality (LID) over samples from smooth-GTID (with $\sigma = 0.5$) using QuickFuzz instance of *SampleGen*.

Based on guidance from the work by Cohen et al. [10], we select $\sigma = 0.5$ for the main experiment, shown for each instance of *SampleGen* in Figure 9. Experimentally, we observed similar results regardless of the choice of σ. The randomized smoothing nullifies any false positives from the Zoo instance, despite losing some coverage in the form of false negatives. With the QuickFuzz instance, some false positives are admitted, although they are rare compared to the prior cases in Section 6.1. We interpret the success of QuickFuzz as a symptom of decreased accuracy, due to the noisy instantiation of *Train*, and the neural network's lack of capacity for such noise. We can infer that choosing a more expressive model is the key to truly certifying GTID. Given the removal of the original GTID heuristics, we can further conclude that existing domain-specific heuristics have little to offer in terms of adversarial robustness.

With the smooth-GTID system, we revisit our original post-mortem analysis using LID, albeit using only the QuickFuzz instance of *SampleGen*, as smooth-GTID thwarts the Zoo attack. The LID scores for the smooth-GTID attack signatures is shown in Figure 10. A main takeaway is that LID scores are either similar to their normal counterparts, or very high. When comparing against Figure 8, there is no visible trend in terms of maximum or minimum LID scores. We do note that signatures with high 'normal' LID score had a high 'adv' LID score, and vice versa for low scoring signatures.

Due to the small sample size, it is difficult to make general claims, but these initial results could suggest that the signature space inhibits high LID as a result of the randomized smoothing mechanism, at the expense of raising the baseline LID score of normal samples.

8 DISCUSSION

From our experiments, we uncover attack surfaces in the algorithms of our formalized MDA system. Each algorithm can be exploited due to different assumptions that are made during design of an MDA system.

In Section 6.2, we saw that QuickFuzz and Zoo only needed a handful of features to fool the MDA systems under test. Since *A* is essentially blind, *SampleGen* allows advancing blindly in input space, through selective fuzzing in the case of QuickFuzz, or with guided heuristics as in Zoo, until a correct combination of brittle features is met. LID analysis in Section 6.3 shows that sub-manifolds exist in the signature space with high complexity. Since signatures are the output of ***SigGen***, we can infer that design of *SigGen* is to blame. This may explain why the adversary does not need many queries. In this sense, we notice some similarity to single-pixel attacks in the image domain [45]. However, we also consider the following. Although all features may contribute towards a learning task, the ***Train*** algorithm conditions the model to attend to certain features. As described by Goodfellow et al. [19], an adversary needs to only find the feature values which the model aligns most with. Despite being less data-driven, MDA systems are equally vulnerable to this phenomenon.

Apart from *SigGen* and *Train*, we showed that previously proposed heuristics for designing ***Auth*** are not secure. Although the decision process can benefit from hand-crafted features, they do not imply robustness against adversaries. The feature attribution analysis showed that target features vary between victims, and even between attacks. Thus, it is not practical to use domain-specific heuristics for defending models. Instead, one must consider underlying properties of adversarial samples. LID is an encouraging first step, as it can abstract away knowledge of specific features. We took this abstraction further, and showed that designers of MDA systems can essentially ignore domain knowledge, by applying end-to-end style defenses such as randomized smoothing. However, such techniques rely on sufficiently expressive models. We summarize by recommending designers focus on applying end-to-end style defenses, which shifts the security challenge to one of model selection, rather than heuristics crafting.

9 RELATED WORK

The progression of attacks in the Adversarial Machine Learning (AML) space often focus on particular applications of machine learning systems. Papernot et al. provide a recent survey of the general deep learning attack landscape, including a high-level view of different threat models and adversary goals [39]. Due to the constrained, hard-label feedback of learning systems in-the-wild, limited-information attacks are incredibly valuable, despite tending to be less powerful. However, such attacks are less prevalent in the literature. Conceptually, the problem can be framed in the restricted query model outlined by Angluin [2]. Biggio briefly considers a concept learning attack against a signature authentication system [5], although in the absence of counting adversarial queries. These iterative attacks evolved into camouflage-style attacks which intend to cloak the adversary in 'fashionably'-crafted accessories or clothing that can fool the authentication system [16, 43]. Recent years have seen the state-of-the-art in limited information attacks, which rely on zeroth-order optimization (ZOO) methods to approximate the victim model's gradient information [7]. The most recent ZOO attacks rely only on the top-1 decision from the victim model [6, 8, 9].

10 CONCLUSION

Although machine learning is a powerful tool for performing device authentication, previous works failed to consider the susceptibility of such systems to AML-style attacks. We demonstrate new restricted-query attacks against device authentication that are successful regardless of the underlying machine learning model. With the help of XAI techniques, we discover that the features used in device authentication are often brittle, and selective perturbation of certain features can be highly effective at breaking device authentication systems.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dave Tian and Scott Clouse for their helpful comments on early revisions of the paper. This work was supported by AFOSR FA9550-19-1-0169, NSF CNS-1815883, and NSF CNS-1562485.

REFERENCES

[1] L. Amsaleg, J. Bailey, D. Barbe, S. Erfani, M. E. Houle, V. Nguyen, and M. Radovanović. 2017. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. 1–6. https://doi.org/10.1109/WIFS.2017.8267651

[2] Dana Angluin. 1988. Queries and Concept Learning. *Machine Learning* 2, 4 (1988), 319–342. https://doi.org/10.1023/A:1022821128753 arXiv:arXiv:1011.1669v3

[3] Marco V. Barbera, Alessandro Epasto, Alessandro Mei, Sokol Kosta, Vasile C. Perta, and Julinda Stefa. 2013. CRAWDAD dataset sapienza/probe-requests (v. 2013-09-10). Downloaded from https://crawdad.org/sapienza/probe-requests/20130910. https://doi.org/10.15783/C76C7Z

[4] Adam Bates, Ryan Leonard, Hannah Pruse, Daniel Lowd, and Kevin R B Butler. 2014. Leveraging USB to Establish Host Identity Using Commodity Devices. *Proceedings of the Network and Distributed System Security (NDSS) Symposium.*

[5] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security Evaluation of Pattern Classifiers under Attack. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 26, 4 (April 2014), 984–996.

[6] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. 2019. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. (April 2019). arXiv:1904.02144 http://arxiv.org/abs/1904.02144

[7] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17* (2017), 15–26. https://doi.org/10.1145/3128572.3140448

[8] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2018. Query-Efficient Hard-label Black-box Attack:An Optimization-based Approach. *arXiv:1807.04457 [cs, stat]* (July 2018). http://arxiv.org/abs/1807.04457 arXiv: 1807.04457.

[9] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. 2020. SIGN-OPT: A QUERY-EFFICIENT HARD-LABEL ADVERSARIAL ATTACK. *The International Conference on Learning Representations (ICLR)* (2020), 16.

[10] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. *arXiv:1902.02918 [cs, stat]* (June 2019). http://arxiv.org/abs/1902.02918 arXiv: 1902.02918.

[11] Asish Kumar Dalai, , and Sanjay Kumar Jena. 2017. WDTF: A Technique for Wireless Device Type Fingerprinting. *Wireless Personal Communications* 97 (2017). https://doi.org/10.1007/s11277-017-4652-y

[12] Boris Danev and Srdjan Capkun. 2009. Transient-based identification of wireless sensor nodes. *IPSN '09 Proceedings of the 2009 International Conference on Information* (2009), 25–36. https://doi.org/10.1145/1602165.1602170

[13] Hung Dang, Yue Huang, and Ee-Chien Chang. 2017. Evading Classifiers by Morphing in the Dark. (2017). https://doi.org/10.1145/3133956.3133978

[14] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher J. Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. 2019. Explanations can be

manipulated and geometry is to blame. *arXiv:1906.07983 [cs, stat]* (June 2019). http://arxiv.org/abs/1906.07983 arXiv: 1906.07983.
[15] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. 2019. Learning Perceptually-Aligned Representations via Adversarial Robustness. *arXiv:1906.00945 [cs, stat]* (June 2019). http://arxiv.org/abs/1906.00945 arXiv: 1906.00945.
[16] Ranran Feng and Balakrishnan Prabhakaran. 2013. Facilitating Fashion Camouflage Art. In *Proceedings of the 21st ACM International Conference on Multimedia (MM '13)*. 793–802. https://doi.org/10.1145/2502081.2502121
[17] Aidin Ferdowsi and Walid Saad. 2018. Deep Learning-Based Dynamic Watermarking for Secure Signal Authentication in the Internet of Things. *IEEE International Conference on Communications* 2018-May (2018). https://doi.org/10.1109/ICC.2018.8422728 arXiv:1711.01306
[18] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 12 pages.
[19] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
[20] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. LEMNA: Explaining Deep Learning based Security Applications. In *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS'18)*.
[21] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 10–18. https://doi.org/10.1145/1656274.1656278
[22] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2019. Natural Adversarial Examples. (2019). arXiv:1907.07174
[23] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. 2019. PassGAN: A Deep Learning Approach for Password Guessing. *arXiv:1709.00440 [cs, stat]* (Feb. 2019). http://arxiv.org/abs/1709.00440 arXiv: 1709.00440.
[24] Michael E. Houle. 2013. Dimensionality, Discriminability, Density and Distance Distributions. In *2013 IEEE 13th International Conference on Data Mining Workshops*. 468–473. https://doi.org/10.1109/ICDMW.2013.139
[25] Jingyu Hua, Hongyi Sun, Zhenyu Shen, Zhiyun Qian, and Sheng Zhong. 2018. Accurate and Efficient Wireless Device Fingerprinting Using Channel State Information. *Proceedings - IEEE INFOCOM* 2018-April (2018), 1700–1708. https://doi.org/10.1109/INFOCOM.2018.8485917
[26] P. A. Johnson, B. Tan, and S. Schuckers. 2010. Multimodal fusion vulnerability to non-zero effort (spoof) imposters. In *2010 IEEE International Workshop on Information Forensics and Security*. 1–5. https://doi.org/10.1109/WIFS.2010.5711469
[27] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. 2015. Evasion and Hardening of Tree Ensemble Classifiers. (2015). arXiv:1509.07892 http://arxiv.org/abs/1509.07892
[28] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. 2005. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing (TDSC)* 2, 2 (April 2005), 93–108.
[29] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. 2018. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. *arXiv:1801.02613 [cs]* (March 2018). http://arxiv.org/abs/1801.02613 arXiv: 1801.02613.
[30] Marwa Mamdouh, Mohamed A.I. Elrukhsi, and Ahmed Khattab. 2018. Securing the Internet of Things and Wireless Sensor Networks via Machine Learning: A Survey. *2018 International Conference on Computer and Applications, ICCA 2018* Section II (2018), 215–218. https://doi.org/10.1109/COMAPP.2018.8460440
[31] Yair Meidan, Michael Bohadana, Asaf Shabtai, Martin Ochoa, Nils Ole Tippenhauer, Juan Davis Guarnizo, and Yuval Elovici. 2017. Detection of Unauthorized IoT Devices Using Machine Learning Techniques. (2017). https://doi.org/10.1002/prot.21521 arXiv:1709.04647
[32] Kevin Merchant, Shauna Revay, George Stantchev, and Bryan Nousain. 2018. Deep Learning for RF Device Fingerprinting in Cognitive Communication Networks. *IEEE Journal on Selected Topics in Signal Processing* 12, 1 (2018), 160–167. https://doi.org/10.1109/JSTSP.2018.2796446
[33] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, Tommaso Frassetto, N Asokan, Ahmad Reza Sadeghi, and Sasu Tarkoma. 2017. IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. In *Proceedings - International Conference on Distributed Computing Systems*. 2511–2514. https://doi.org/10.1109/ICDCS.2017.284 arXiv:1611.04880
[34] Shervin Minaee and Amirali Abdolrashidi. 2018. Finger-GAN: Generating Realistic Fingerprint Images Using Connectivity Imposed GAN. *arXiv:1812.10482 [cs]* (Dec. 2018). http://arxiv.org/abs/1812.10482 arXiv: 1812.10482.
[35] Yurii Nesterov and Vladimir Spokoiny. 2017. Random Gradient-Free Minimization of Convex Functions. *Foundations of Computational Mathematics* 17, 2 (April 2017), 527–566. https://doi.org/10.1007/s10208-015-9296-2
[36] Nicholas Papernot and contributors. 2019. Cleverhans. https://github.com/tensorflow/cleverhans.
[37] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIACCS)*. 14 pages.
[38] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *Proceedings of the IEEE European Symposium on Security and Privacy (Euro S&P)*. 372–387. https://doi.org/10.1109/EuroSP.2016.36
[39] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. 399–414. https://doi.org/10.1109/EuroSP.2018.00035
[40] Sakthi Vignesh Radhakrishnan, A. Selcuk Uluagac, and Raheem Beyah. 2015. GTID: A Technique for Physical Device and Device Type Fingerprinting. *IEEE Transactions on Dependable and Secure Computing* 12, 5 (2015), 519–532. https://doi.org/10.1109/TDSC.2014.2369033
[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.
[42] Ricardo N. Rodrigues, Lee Luan Ling, and Venu Govindaraju. 2009. Robustness of multimodal biometric fusion methods against spoof attacks. *Journal of Visual Languages & Computing* 20, 3 (2009), 169 – 179. https://doi.org/10.1016/j.jvlc.2009.01.010 ADVANCES IN MULTIMODAL BIOMETRIC SYSTEMS.
[43] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[44] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2019. How can we fool LIME and SHAP? Adversarial Attacks on Post hoc Explanation Methods. (Nov. 2019). arXiv:1911.02508 http://arxiv.org/abs/1911.02508
[45] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. 2017. One pixel attack for fooling deep neural networks. (2017). arXiv:1710.08864 http://arxiv.org/abs/1710.08864
[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Porceedings of the International Conference on Learning Representations (ICLR)*.
[47] A. Selcuk Uluagac. 2014. CRAWDAD dataset gatech/fingerprinting (v. 2014-06-09). Downloaded from https://crawdad.org/gatech/fingerprinting/20140609. https://doi.org/10.15783/C78G67
[48] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. 2019. Interpretable Deep Learning under Fire. https://arxiv.org/abs/1812.00891
[49] Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Mohamed Ali Kaafar. 2020. On the Resilience of Biometric Authentication Systems against Random Inputs. *Proceedings 2020 Network and Distributed System Security Symposium* (2020). https://doi.org/10.14722/ndss.2020.24210 arXiv: 2001.04056.

A APPENDIX

A.1 Implementation Details

USB-Fingerprinting. We implemented *Auth* in Python using a wrapper[4] for Weka's RandomForest classifier [21]. We used model hyperparameters, Weka Java archive file, and trace data provided by the authors. In order to generate the training data, we used the trace data of nine computers of identical make and model.

GTID. We implemented GTID in Python using Tensorflow to compile and train the ANNs. *Train* uses Adam optimizer and categorical cross-entropy loss over 60 epochs to output the trained model C for *Auth* to use. We processed the data for training the ANNs from the author's GTID dataset hosted on CRAWDAD [47]. For experiments, we used the Iperf–TCP traffic of their *isolatedTestBed* set, and mimicked the authors' published hyperparameters.

WDTF. We implemented WDTF in Python using NumPy. We obtained the training data from a collection of wireless probe requests hosted on CRAWDAD (same data source as used in the original work [3]). Note that we use WDTF primarily to compare MDA systems that use customized, statistical-distance-based model versus systems that use traditional ML models.

[4] https://github.com/fracpete/python-weka-wrapper3