PassGAN: A Deep Learning Approach for Password Guessing

Briland Hitaj*, Paolo Gasti†, Giuseppe Ateniese* and Fernando Perez-Cruz‡

*Computer Science Department, Stevens Institute of Technology
Email: {bhitaj, gatenies}@stevens.edu

†Computer Science Department, New York Institute of Technology
Email: pgasti@nyit.edu

‡Swiss Data Science Center, (ETH Zurich and EPFL)
Email: fernando.perezcruz@sdsc.ethz.ch

***Abstract*—State-of-the-art password guessing tools, such as HashCat and John the Ripper, enable users to check billions of passwords per second against password hashes. In addition to performing straightforward dictionary attacks, these tools can expand password dictionaries using password generation rules, such as concatenation of words (e.g., "password123456") and *leet speak* (e.g., "password" becomes "p4s5w0rd"). Although these rules work well in practice, expanding them to model further passwords is a laborious task that requires specialized expertise.**

To address this issue, in this paper we introduce PassGAN, a novel approach that replaces human-generated password rules with theory-grounded machine learning algorithms. Instead of relying on manual password analysis, PassGAN uses a Generative Adversarial Network (GAN) to autonomously learn the distribution of real passwords from actual password leaks, and to generate high-quality password guesses. Our experiments show that this approach is very promising. When we evaluated PassGAN on two large password datasets, we were able to surpass rule-based and state-of-the-art machine learning password guessing tools. However, in contrast with the other tools, PassGAN achieved this result without any a-priori knowledge on passwords or common password structures. Additionally, when we combined the output of PassGAN with the output of HashCat, we were able to match 51%-73% more passwords than with HashCat alone. This is remarkable, because it shows that PassGAN can autonomously extract a considerable number of password properties that current state-of-the art rules do not encode.

I. Introduction

Passwords are the most popular authentication method, mainly because they are easy to implement, require no special hardware or software, and are familiar to users and developers. Unfortunately, multiple password database leaks have shown that users tend to choose easy-to-guess passwords [18], [24], [54], primarily composed of common strings (e.g., `password`, `123456`, `iloveyou`), and variants thereof.

Password guessing tools provide a valuable tool for identifying weak passwords, especially when they are stored in hashed form [68], [73]. The effectiveness of password guessing software relies on the ability to quickly test a large number of highly likely passwords against each password hash. Instead of exhaustively trying all possible character combinations, password guessing tools use words from dictionaries and previous password leaks as candidate passwords. State-of-the-art password guessing tools, such as John the Ripper [86] and HashCat [36], take this approach one step further by defining heuristics for password transformations, which include combinations of multiple words (e.g., `iloveyou123456`), mixed letter case (e.g., `iLoVeyOu`), and *leet speak* (e.g., `il0v3you`). These heuristics, in conjunction with Markov models, allow John the Ripper and HashCat to generate a large number of *new* highly likely passwords.

While these heuristics are reasonably successful in practice, they are ad-hoc and based on intuitions on how users choose passwords, rather than being constructed from a principled analysis of large password datasets. For this reason, each technique is ultimately limited to capturing a specific subset of the password space which depends upon the intuition behind that technique. Further, developing and testing new rules and heuristics is a time-consuming task that requires specialized expertise, and therefore has limited scalability.

A. *Our Approach*

To address these shortcomings, in this paper we propose to replace rule-based password guessing, as well as password guessing based on simple data-driven techniques such as Markov models, with a novel approach based on deep learning. At its core, our idea is to train a neural networks to autonomously determine password characteristics and structures, and to leverage this knowledge to generate new samples that follow the same distribution. We hypothesize that deep neural networks are expressive enough to capture a large variety of properties and structures that describe the majority of user-chosen passwords; at the same time, neural networks can be trained without any a-priori knowledge or assumption on such properties and structures. This is in stark contrast with current approaches such as Markov models (which implicitly assume that all relevant password characteristics can be defined in terms of n-grams), and rule-based approaches (which can guess only passwords that match with the available rules). As a result, samples generated using a neural network are not limited to a particular subset of the password space. Instead, neural networks can autonomously encode a wide range of password-guessing knowledge that includes and surpasses what is captured in human-generated rules and Markovian password generation processes.

To test this hypothesis, in this paper we introduce PassGAN, a new approach for generating password guesses based on deep learning and Generative Adversarial Networks (GANs) [31]. GANs are recently-introduced machine-learning tools designed to perform density estimation in high-dimensional spaces [31]. GANs perform implicit generative modeling by training a deep neural network architecture that is fed a simple random distribution (e.g., Gaussian or uniform) and by generating samples that follow the distribution of the available data. In a way, they implicitly model $\mathbf{x} = F^{-1}(s)$ where $F(\mathbf{x})$ is the cumulative density function of the data and s is a uniformly distributed random variable. To train the generative structure, GANs use a cat-and-mouse game in which a deep generative network (G) tries to mimic the underlying distribution of the samples and a discriminative deep neural network (D) tries to distinguish between the original training samples (i.e., "true samples") and the samples generated by G (i.e., "fake samples"). This adversarial procedure forces D into leaking the relevant information for G to be effective at mimicking the original distribution of the data.

PassGAN leverages this technique to generate new password guesses. We train D using a list of leaked passwords (real samples), see Figure 1. Therefore, at each iteration, the output of PassGAN (fake samples) becomes closer to the distribution of passwords in the original leak, and hence more likely to match real users' passwords. To the best of our knowledge, this work is the first to use GANs for this purpose.

B. Contributions

PassGAN represents a principled and theory-grounded take on the generation of password guesses. We explore and evaluate different neural network configurations, parameters, and training procedures, to identify the appropriate balance between *learning* and *overfitting*, and report our results. Specifically, our contributions are as follows:

1) We show that a properly-trained GAN can generate high-quality password guesses. Our password generation GAN is trained on a portion of RockYou dataset and tested on two different datasets: (1) another (distinct) fraction of RockYou dataset and (2) a dataset of leaked passwords from LinkedIn. Details of this test are in Section IV-A. In our experiments, we were able to match 1,350,178 (43.6%) of 3,094,199 *unique* passwords from a testing set composed of real user passwords from the RockYou dataset [76], and 10,478,322 (24.2%) out of 43,354,871 passwords from the LinkedIn dataset. To verify the ability of PassGAN to create unseen passwords, we removed from each testing dataset those password samples that are present in the training set. This operation resulted in 1,978,367 and 40,593,536 unique password subsets remaining in RockYou and LinkedIn testing sets respectively. Out of these, PassGAN was able to match 676,439 (34.6%) of samples in RockYou dataset and 8,878,284 (34.2%) of samples in LinkedIn.
Moreover, the overwhelming majority of passwords generated by PassGAN that did not match our testing set still "looked like" human-generated passwords, and thus could potentially match real user accounts not considered in our experiments.
2) We show that PassGAN is competitive with state-of-the-art password generation rules. Even though these rules were specifically tuned for the datasets used in our evaluation, the quality of PassGAN's output was comparable to that of password rules.
3) With password generation rules, the number of unique passwords that can be generated is defined by the number of rules and by the size of the password dataset used to instantiate them. In contrast, PassGAN can output a practically unbounded number of password guesses. Crucially, our experiments show that with PassGAN the number of matches increases steadily with the number of passwords generated. This is important, because it shows that the output of PassGAN is not restricted to a small subset of the password space. As a result, in our experiments PassGAN was able to eventually guess more passwords than any of the other tools, even though all tools were trained on the same password dataset. This, however, came at the cost of generating a larger number of passwords with PassGAN than with the other tools.
4) PassGAN is competitive with the current state of the art in password guessing algorithms based on neural networks [58]. Our results show that PassGAN essentially matches the performance of Melicher et al. [58] (indicated as FLA in the rest of the paper). While PassGAN achieved this result using a larger number of passwords, it also correctly guessed passwords with structures that FLA was unable to model.
5) We show that PassGAN can be effectively used to *complement* password generation rules. In our experiments, PassGAN matched password that were not generated by any password rule. When we combined the output of PassGAN with the output of HashCat, we were able to guess between 51% and 73% additional unique passwords compared to HashCat alone.

We consider this work the first step towards fully automated generation of high-quality password guesses. Currently, there is a tradeoff between the benefits of PassGAN (i.e., expressiveness, generality, and ability to autonomously learn from samples), and its cost in terms of output size, compared to rule-based approaches. While rule-based password guessing tools are able to generate a significant number of matches within a remarkably small number of attempts, PassGAN must output a larger number of passwords to achieve the same result. We argue that this is, in practice, not an issue because: (1) password guessing tools can be easily combined in such a way that once a fast tool (e.g., HashCat) has exhausted its attempts, a more comprehensive one (such as PassGAN) can continue to generate new matches; and (2) the cost of storage has been steadily decreasing for decades, to the point that a cheap 256 GB USB drive can store more than 10^{10} password guesses. As such, password generation can be treated as an offline process.

For these reasons, we believe that meaningful comparisons between password guessing techniques should primarily focus on the number of matches that each technique is able to generate, rather than *how quickly* these matches are generated.

We argue that this work is relevant, important, and timely. *Relevant*, because despite numerous alternatives [69], [82], [27], [23], [96], we see little evidence that passwords will be replaced any time soon. *Important*, because establishing the limits of password guessing—and better understanding how guessable real-world passwords are—will help make password-based systems more secure. And *timely*, because recent leaks containing hundreds of millions of passwords [26] provide a formidable source of data for attackers to compromise systems, and for system administrators to re-evaluate password policies.

C. PassGAN in the Media and in Academia

The ability of PassGAN to autonomously learn characteristics and patterns constituting a password, much like DeepMinds' AlphaGo's ability to autonomously learn the game of Go [3], drew significant attention from several media outlets. For instance, PassGAN has been reported in articles from Science Magazine [42], The Register [88], Inverse [56], Threatpost [60], Dark Reading [91], Technology Review News [16], Sensors Online [20], and others [71], [43], [14], [55], [29]. Further, PassGAN was selected by Dark Reading as one of *the coolest hacks of 2017* [39].

UC Berkeley has included PassGAN as reading material in their graduate-level course titled *Special Topics in Deep Learning* [5].

D. Changes with Respect to an Earlier Version of this Paper

This paper updates and extends an earlier version of our work [41]. The differences between the two versions of the paper can be summarized as follows: (1) we identified an issue with the PassGAN implementation used in [41], which led to a substantial decrease in the number of unique passwords generated. We corrected this issue and, as a result, in this paper we report a rate of generation of unique passwords roughly four times higher than in our earlier work; and (2) in the updated paper, we compare PassGAN with state-of-the-art password guessing based on Markov Models, and with the work on neural-network (RNN) by Melicher et al. [58], in addition to John the Ripper and HashCat.

E. Organization

The rest of this paper is organized as follows. In Section II, we briefly overview deep learning, GANs, and password guessing, and provide a summary of the relevant state of the art. Section III discusses the architectural and training choices for the GAN used to instantiate PassGAN, and the hyperparameters used in our evaluation. We report on the evaluation of PassGAN, and on the comparison with state-of-the-art password guessing techniques, in Section IV. We summarize our findings, and discuss their implications, in Section V. We conclude in Section VI.

II. Background and Related Work

In this section, we present a brief overview of deep learning and GANs. We then review the state of the art in password guessing.

A. Deep Learning

In the mid-nineties, machine learning methods such as support vector machines [79], random forests [9], and Gaussian processes [75], showed remarkable results in classification and regression for mostly uncorrelated human engineered (hand-coded) features. Starting in the mid-2000s, with an increased availability of storage and data, these methods have been superseded by deep learning. Research on deep learning has shown that features can effectively be learned from data, and that hand-coded features tend to underperform learned features. These gains are more relevant with correlated features, in which human-engineered features might only encode low-dimensional correlations.

Deep learning has been extensively used in various areas of computer science, including computer vision [51], image processing [85], video processing [21], [65], speech recognition [34], natural language processing [2], [15], [95], and gaming [32], [48], [59], [62]. Further, deep learning has led to significant advances in other areas, including healthcare [17], [25]. In the areas of privacy and security, research on deep learning has focused on three major thrusts: (1) privacy-preserving deep learning [80] and differentially-private deep learning [1]; (2) attacks on deep learning models that are trained on private data [6], [28], [81], [37], [90], [40]; and (3) attacks that lead to input misclassifications for otherwise very accurate models [67], [66], [57], [10], [11], [46], [53], [38].

B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) represent a remarkable advance in the area of deep learning. A GAN is composed of two neural networks, a generative deep neural network G, and a discriminative deep neural network D. Given an input dataset $\mathcal{I} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the goal of G is to produce "fake" samples from the underlying probability distribution $\Pr(\mathbf{x})$, that are accepted by D. At the same time, D's goal is to learn to distinguish fake samples from G from the real ones coming from $\mathcal{I}$. More formally, on input a simple noise distribution $\mathbf{z}$, the optimization problem solved by GANs can be summarized as follows:

$$\min_{\theta_G} \max_{\theta_D} \sum_{i=1}^{n} \log f(\mathbf{x}_i; \theta_D) + \sum_{j=1}^{n} \log(1 - f(g(\mathbf{z}_j; \theta_G); \theta_D))$$

where the model attempts to minimize with respect to θ_G, and simultaneously maximize with respect to θ_D. The learning phase is considered complete when D is unable to distinguish between the fake samples produced by G, and the real samples from $\mathcal{I}$.

Since the original work by Goodfellow et al. [31], there have been several improvements on GANs. Radford et al. [74] introduce DCGAN, which improves on [31] by using a convolutional neural network instead of a multi-layer perceptron. As a result, DCGAN can produce more realistic image samples compared to [31].

Other work on GANs includes BEGAN [7], DiscoGAN [44], Conditional GAN [61], AdaGAN [89], InfoGAN [13], Laplacian Pyramid GAN [19], and StackGAN [94]. These techniques introduce improvements to prior work, such as new approaches to training and using the GAN.

Arjovsky et al. [4] introduced the Wasserstein GAN (WGAN). WGAN improves learning stability of prior GANs by using gradient clipping. Benefits of this approach include reduced mode collapse, and meaningful learning curves, which are helpful in identifying optimal hyperparameters.

All work above focuses on the generation of realistic images. To address the problem of text generation, Gulrajani et al. [35] recently introduced the Improved Wasserstein GAN (IWGAN). With IWGAN, both G and D are simple convolutional neural networks (CNNs). G takes as input a latent noise vector, transforms it by forwarding it through its convolutional layers, and outputs a sequence of 32 one-hot character vectors. A *softmax* nonlinearity is applied at the output of G, and then forwarded to D. Each output character from IWGAN is obtained by computing the *argmax* of each output vector produced by G. Figure 1 shows the two main components of a GAN (i.e., G and D) with their respective inputs and outputs.

C. Password Guessing

In a password guessing attack, the adversary attempts to identify the password of one or more users by repeatedly testing multiple candidate passwords. Password guessing attacks are probably as old as password themselves [8], with more formal studies dating back to 1979 [63].

Two popular modern password guessing tools are John the Ripper (JTR) [86] and HashCat [36]. Both tools implement multiple types of password guessing strategies, including: exhaustive brute-force attacks; dictionary-based attacks; rule-based attacks, which consist in generating password guesses from transformations of dictionary words [78], [77]; and Markov-model-based attacks [87], [70], in which each character of a password is selected via a stochastic process that considers one or more preceding character, and which is trained on dictionaries of plaintext passwords. JTR and HashCat are notably effective at guessing passwords. Specifically, there have been several instances in which well over 90% of the password leaked from online services have been successfully recovered [72].

Markov models were first used to generate password guesses by Narayanan et al. [64]. Their approach uses manually defined password rules, such as which portion of the generated passwords is composed of letters and numbers. This technique was subsequently improved by Weir et al. [92], who showed how to "learn" these rules from password distributions. This early work has been subsequently extended by Ma et al. [54] and by Durmuth et al. [24]. Techniques based on Markov models have also been used to implement real-time password strength estimators, and to evaluate the strength of passwords in plaintext databases (see, e.g., [18], [12]).

Recently, Melicher et al. [58] introduced FLA, a password guessing method based on recurrent neural networks [33], [84]. With this technique, the neural network is trained using passwords leaked from several websites. During password generation, the neural network outputs one password character at a time. Each new character (including a special end-of-password character) is selected based on its probability, given the current output state, in what is essentially a Markov process. (This property was leveraged in [58] primarily to perform real-time password strength estimation.) The primary goal of Melicher et al. [58] with FLA is to provide fast and accurate password strength estimation while keeping the model as lightweight as possible, and without sacrificing accuracy. As such, given a trained FLA model, one can feed a set of passwords to the model and retrieve as output a file containing 6 fields organized as follows: (1) password, (2) the probability of that password, (3) the estimated output guess number, i.e., the strength of that password, (4) the std deviation of the randomized trial for this password (in units of number of guesses), (5) the number of measurements for this password, and (6) the estimated confidence interval for the guess number (in units of number of guesses). The evaluation presented in [58] shows that their technique outperforms Markov models and password composition rules commonly used with JTR and HashCat, when testing a large number of password guesses (in the 10^{10} to 10^{25} range). In terms of expressiveness, we argue that the differences between the output probabilities of FLA and PassGAN are due to the Markovian structure of the password generation process in FLA. Because of this property, any password characteristic that is not captured within the scope of an n-gram might not be encoded by FLA. For instance, if a meaningful subset of 10-character passwords are constructed as the concatenation of two words (e.g., `MusicMusic`), any Markov process with $n \leq 5$ will not be able to properly capture this behavior. On the other hand, given enough examples, the neural network used in PassGAN will be able to learn this property. As result, while password `pookypooky` was assigned a probability $p \approx 10^{-33}$ by FLA (with an estimated number of guessing attempts of about 10^{29}), it was guessed after roughly 10^{8} attempts by PassGAN.

III. GAN Architecture and Hyperparameters

To leverage the ability of GANs to effectively estimate the probability distribution of passwords from the training set, we experimented with a variety of parameters. In this section, we report our choices on specific GAN architecture and hyperparameters.

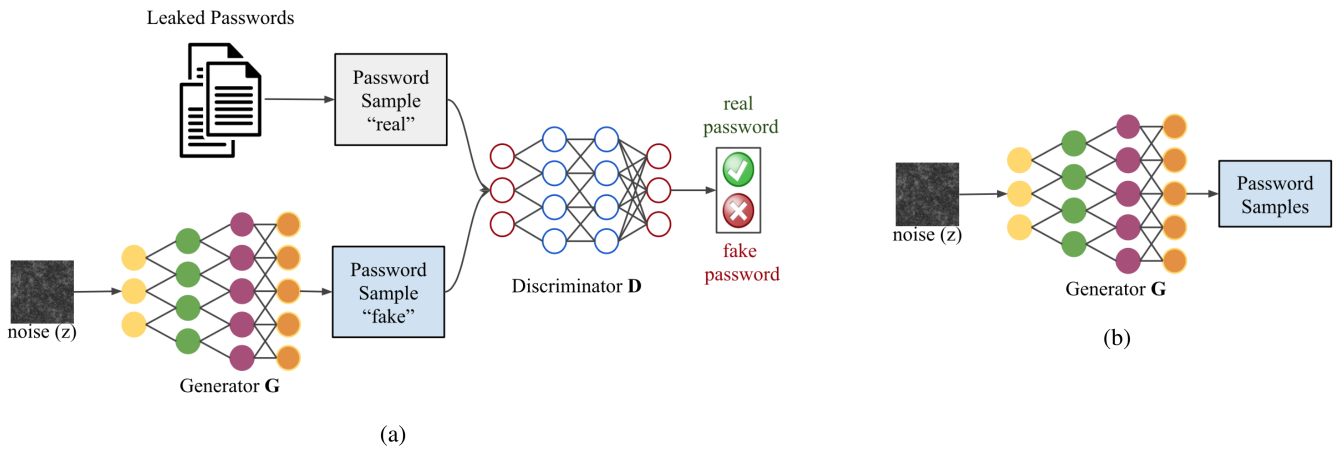

Fig. 1: Summary of PassGAN's Architecture. In the training procedure, shown in (a), the discriminator (D) processes passwords from the training dataset, as well as password samples produced by the generator (G). Based on the feedback from D, G fine-tunes its network to produce password samples that are close to the training set (G has no direct access to the training set). The password generation procedure is shown in (b).

We instantiated PassGAN using the *Improved training of Wasserstein GANs* (IWGAN) of Gulrajani et al. [35]. The IWGAN implementation used in this paper relies on the ADAM optimizer [45] to minimize the training error, i.e., to reduce the mismatch between the output of the model and its training data.

Our model is characterized by the following hyper-parameters:

- **Batch size**, which represents the number of passwords from the training set that propagate through the GAN at each step of the optimizer.
- **Number of iterations**, which indicates how many times the GAN invokes its forward step and its back-propagation step [50], [49]. In each iteration, the GAN runs one generator iteration and one or more discriminator iterations.
- **Number of discriminator iterations per generator iteration**, which indicates how many iterations the generator performs in each GAN iteration.
- **Model dimensionality**, which represents the number of dimensions (weights) for each convolutional layer.
- **Gradient penalty coefficient** (λ), which specifies the penalty applied to the norm of the gradient of the discriminator with respect to its input [35]. Increasing this parameter leads to a more stable training of the GAN [35].
- **Output sequence length**, which indicates the maximum length of the strings generated by the generator (G henceforth).
- **Size of the input noise vector (seed)**, which determines how many random bits are fed as input to G for the purpose of generating samples.
- **Maximum number of examples**, which represents the maximum number of training items (passwords, in the case of PassGAN) to load.
- **Adam optimizer's hyper-parameters**:
  - **Learning rate**, i.e., how quickly the weights of the model are adjusted
  - **Coefficient** β_1, which specifies the decaying rate of the running average of the gradient.
  - **Coefficient** β_2, which indicates the decaying rate of the running average of the square of the gradient.

We instantiated our model with a *batch size* of 64. We trained the GAN using various *number of iterations* and eventually settled for 199,000 iterations, as further iterations provided diminishing returns in the number of matches (see analysis in Section IV-A). The *number of discriminator iterations per generative iteration* was set to 10, which is the default value used by IWGAN. We experimented using 5 residual layers for both the generator and the discriminator, with each of the layers in both deep neural network having 128 *dimensions*.

We set the *gradient penalty* to 10 and modified the *length of the sequence generated by the GAN* from 32 characters (default length for IWGAN) to 10 characters, to match the maximum length of passwords used during training (see Section IV-A). The *maximum number of examples* loaded by the GAN was set to the size of the entire training dataset. We set the *size of the noise vector* to 128 floating point numbers.

Coefficients β_1 and β_2 of the Adam optimizer were set to 0.5 and 0.9, respectively, while the learning rate was 10^{-4}. These parameters are the default values used by Gulrajani et al. [35].

IV. Evaluation

In this section, we first present our training and testing procedures. We then quantify PassGAN's ability to guess passwords, and we compare it with FLA, with a popular 3-gram implementation of Markov models [22], and with password generation rules for JTR (SpiderLab mangling rules [83]) and HashCat (Best64 and gen2 rules [36]). These password generation rules are commonly used in the password guessing literature (see, e.g., [58]), and have been optimized over the years on password datasets including RockYou and LinkedIn. Because of these dataset-specific optimizations, we consider these rules a good representation of the best matching performance that can be obtained with rules-based password guessing. We conclude this section with an analysis of the output of PassGAN in terms of probability densities and password distribution.

Experiment Setup. Our experiments were run using the TensorFlow implementation of IWGAN. We used TensorFlow version 1.2.1 for GPUs, with Python version 2.7.12. All experiments were performed on a workstation running Ubuntu 16.04.2 LTS, with 64GB of RAM, a 12-core 2.0 GHz Intel Xeon CPU, and an NVIDIA GeForce GTX 1080 Ti GPU with 11GB of global memory.

A. GAN Training and Testing

To evaluate the performance of PassGAN, and to compare it with state-of-the-art password generation rules, we first trained the GAN, as well as JTR, HashCat, the Markov model, and FLA on a large set of passwords from the RockYou password leak [76].[1] Entries in this dataset represent a mixture of common and complex passwords: because they were stored on servers in plaintext, passwords of all complexities were recovered. We then counted how many of the passwords generated by each password guessing tool were present in two separate testing sets: a subset of RockYou distinct from the training set, and the LinkedIn password dataset [52]. Details on these procedures and datasets are presented next.

The RockYou dataset [76] contains 32,503,388 passwords. We selected all passwords of length 10 characters or less (29,599,680 passwords, which correspond to 90.8% of the dataset), and used 80% of them (23,679,744 total passwords, 9,926,278 unique passwords) to train each password guessing tool. For testing, we computed the difference between the remaining 20% of the dataset (5,919,936 total passwords, 3,094,199 unique passwords) and the training test. The resulting 1,978,367 entries correspond to passwords that were not previously observed by the password guessing tools. This allowed us to count only non-trivial matches in the testing set.

We also tested each tool on passwords from the LinkedIn dataset [52], of length up to 10 characters, and that were not present in the training set. The LinkedIn dataset consists of 60,065,486 total unique passwords (43,354,871 unique passwords with length 10 characters or less), out of which 40,593,536 were not in the training dataset. (Frequency counts were not available for the LinkedIn dataset.) Passwords in the LinkedIn dataset were exfiltrated as hashes, rather than in plaintext. As such, the LinkedIn dataset contains only plaintext passwords that tools such as JTR and HashCat were able to recover, thus giving rule-based systems a potential edge.

[1]We consider the use of publicly available password datasets to be ethical, and consistent with security research best practices (see, e.g., [18], [58], [12]).

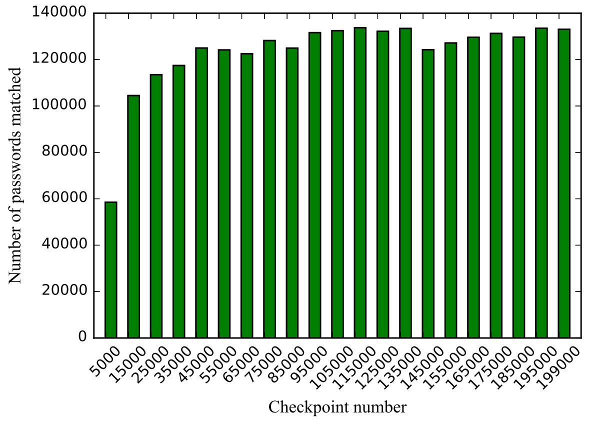

Fig. 2: Number of unique passwords generated by PassGAN on various checkpoints, matching the RockYou testing set. The x axis represents the number of iterations (checkpoints) of PassGAN's training process. For each checkpoint, we sampled 10^8 passwords from PassGAN.

Our training and testing procedures allowed us to determine: (1) how well PassGAN predicts passwords when trained and tested on the same password distribution (i.e., when using the RockYou dataset for both training and testing); and (2) whether PassGAN generalizes across password datasets, i.e., how it performs when trained on the RockYou dataset, and tested on the LinkedIn dataset.

Impact of Training Process on Overfitting. Training a GAN is an iterative process that consists of a large number of iterations. As the number of iterations increases, the GAN learns more information from the distribution of the data. However, increasing the number of steps also increases the probability of overfitting [31], [93].

To evaluate this tradeoff on password data, we stored intermediate training checkpoints and generated 10^8 passwords at each checkpoint. Figure 2 shows how many of these passwords match with the content of the RockYou testing set. In general, the number of matches increases with the number of iterations. This increase tapers off around 125,000-135,000 iterations, and then again around 190,000-195,000 iterations, where we stopped training the GAN. This indicates that further increasing the number of iterations will likely lead to overfitting, thus reducing the ability of the GAN to generate a wide variety of highly likely passwords. Therefore, we consider this range of iterations adequate for the RockYou training set.

B. Evaluating the Passwords Generated by PassGAN

To evaluate the quality of the output of PassGAN, we generated $5 \cdot 10^{10}$ passwords, out of which roughly $7 \cdot 10^9$ were unique. We compared these passwords with the outputs of length 10 characters or less from HashCat Best64 and gen2, JTR SpiderLab, FLA, and Markov model. These passwords were computed as follows:

- We instantiated HashCat and JTR's rules using passwords from the training set sorted by frequency in descending order (as in [58]). HashCat Best64 generated 754,315,842 passwords, out of which 361,728,683 were unique and of length 10 characters or less. With HashCat gen2 and JTR SpiderLab we uniformly sampled a random subset of size 10^9 from their output. This subset was composed of passwords of length 10 characters or less.
- For FLA, we set up the code from [47] according to the instruction provided at [30]. We trained a model containing 2-hidden layers and 1 dense layer of size 512 (for the full list of parameters see Table VIII in Appendix A). We did not perform any transformation (e.g., removing symbols, or transforming all characters to lowercase) on the training set for the sake of consistency with the other tools. Once trained, FLA enumerates a subset of its output space defined by a probability threshold p. A password belongs to FLA's output only if its probability is at least p. In our experiments, we set $p = 10^{-10}$. This resulted in a total of 747,542,984 passwords of length 10 characters or less. Before using these passwords in our evaluation, we sorted them by probability in descending order.
- We generated 494,369,794 unique passwords of length 10 or less using the 3-gram Markov model. We ran this model using its standard configuration [22].

In our comparison, we aimed at establishing whether PassGAN was able to meet the performance of the other tools, despite its lack of any a-priori knowledge on password structures. This is because we are primarily interested in determining whether the properties that PassGAN autonomously extracts from a list of passwords can represent enough information to compete with state-of-the-art human-generated rules and Markovian password generation processes.

Our results show that, for each of the tools, PassGAN was able to generate at least the same number of matches. Additionally, to achieve this result, PassGAN needed to generate a number of passwords that was within one order of magnitude of each of the other tools. This holds for both the RockYou and the LinkedIn testing set. Table I summarizes our findings for the RockYou testing set, while Table II shows our results for the LinkedIn test set.

Our results also show that PassGAN has an advantage with respect to rule-based password matching when guessing passwords from a dataset different from the one it was trained on. In particular, PassGAN was able to match more passwords than HashCat and JTR within a smaller number of attempts ($3.6 \cdot 10^9$ for LinkedIn, compared to $4.8 \cdot 10^9$ – $5.06 \cdot 10^9$ with RockYou for HashCat, and the same number of attempts for JTR, but with a more significant margin in the LinkedIn dataset). This is important, because it is typically the closest setting to how these tools are going to be used in practice.

Combining the Output of Multiple Password Guessing Tools. We then focused our evaluation on the how combining the output of rule-based with machine-learning-based password guessing tools affected guessing performance. Our hypothesis was that, although rule-based tools are fast and effective when guessing passwords that follow the rules on which these tools rely, machine learning tools might be able to match additional passwords, at the cost of a larger number of attempts. To test this hypothesis, we removed all passwords matched by HashCat Best64 (the best performing set of rules in our experiments) from the RockYou and LinkedIn testing sets. This led to a two new test sets, containing 1,348,300 (RockYou) and 33,394,178 (LinkedIn) passwords. We then calculated how many additional matches PassGAN and FLA were able to achieve alone, and combined. Further, we determined how many passwords contributed by PassGAN were not generated by FLA, and vice versa. This provides a direct indication of how close the outputs of these tool are.

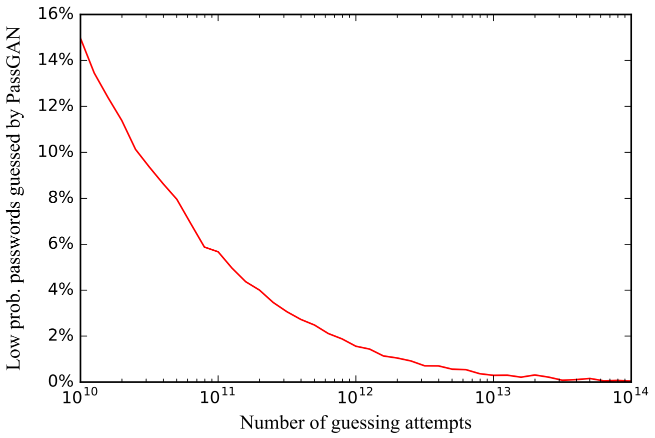

Fig. 3: Percentage of passwords matched by FLA at a particular number of guesses, that are matched by PassGAN in at most $7 \cdot 10^{10}$ attempts.

We report our results with the RockYou testing set in Table III, and with the LinkedIn testing set in Table IV. Our results show that the number of matches generated by each tool steadily increases with their output size. In particular when we used $7 \cdot 10^9$ passwords from PassGAN, we were able to match an additional 51% of passwords from the RockYou dataset, and 73% additional passwords from LinkedIn, compared to HashCat alone. This confirms that both PassGAN and FLA are capable of generating the same number of guesses as rule-based password guessing, and that the passwords that each tool generates are distinct. Therefore, our results indicate that combining rules with machine learning password guessing is an effective strategy. Further, our results show that PassGAN and FLA are able to complement each other well. Specifically, when their output is combined, they both contribute matches. This is confirmed by Columns (4) and (5) in tables III and IV, which show that the overlapping between the output of the two tools is limited. This confirms that PassGAN and FLA are able to capture different portions of the password space.

To investigate further on the differences between PassGAN and FLA, we computed the number of passwords in the RockYou testing set that PassGAN was able to guess within

TABLE I: Number of matches generated by each password guessing tool against the RockYou testing set, and corresponding number of password generated by PassGAN to outperform each tool. Matches for HashCat Best64 and FLA were obtained by exhaustively enumerating the entire output of each tool. The minimum probability threshold for FLA was set to $p = 10^{-10}$.

Approach	(1) Unique Passwords	(2) Matches	(3) Number of passwords required for PassGAN to outperform (2)	(4) PassGAN Matches
JTR Spyderlab	10^9	461,395 (23.32%)	$2.1 \cdot 10^9$	515,079 (26.04%)
Markov Model 3-gram	$4.9 \cdot 10^8$	532,961 (26.93%)	$3.2 \cdot 10^9$	568,135 (28.72%)
HashCat gen2	10^9	597,899 (30.22%)	$4.8 \cdot 10^9$	625,245 (31.60%)
HashCat Best64	$3.6 \cdot 10^8$	630,068 (31.84%)	$5.06 \cdot 10^9$	630,335 (31.86%)
FLA 10^{-10}	$7.4 \cdot 10^8$	652,585 (32.99%)	$6 \cdot 10^9$	653,978 (33.06%)

TABLE II: Number of matches generated by each password guessing tool against the LinkedIn testing set, and corresponding number of password generated by PassGAN to outperform each tool. Matches for HashCat Best64 and FLA were obtained by exhaustively enumerating the entire output of each tool. The minimum probability threshold for FLA was set to $p = 10^{-10}$.

Approach	(1) Unique Passwords	(2) Matches	(3) Number of passwords required for PassGAN to outperform (2)	(4) PassGAN Matches
JTR Spyderlab	10^9	6,840,797 (16.85%)	$3.6 \cdot 10^9$	7,419,248 (18.27%)
Markov Model 3-gram	$4.9 \cdot 10^8$	5,829,786 (14.36%)	$3.6 \cdot 10^9$	7,419,248 (18.27%)
HashCat gen2	10^9	6,308,515 (15.54%)	$3.6 \cdot 10^9$	7,419,248 (18.27%)
HashCat Best64	$3.6 \cdot 10^8$	7,174,990 (17.67%)	$3.6 \cdot 10^9$	7,419,248 (18.27%)
FLA 10^{-10}	$7.4 \cdot 10^8$	8,290,173 (20.42%)	$6 \cdot 10^9$	8,519,060 (21.00%)

TABLE III: Matches generated by PassGAN and FLA in addition to the matches from HashCat Best64. The testing set used in this table was obtained by removing all passwords generated by HashCat Best64 from the RockYou testing set (1,348,300 passwords). Column (4) represent the number of matches obtained with the passwords generated by PassGAN minus the passwords generated by FLA (and vice versa, in Column (5)).

Unique Passwords	(1) PassGAN	(2) FLA	(3) PassGAN ∪ FLA	(4) PassGAN, and not from FLA	(5) FLA, and not from PassGAN
10^4	14	2	16	14	2
10^5	95	40	133	93	38
10^6	881	1,183	2,016	833	1,135
10^7	7,633	16,330	22,203	5,873	14,570
10^8	44,490	117,262	137,415	20,153	92,925
10^9	155,369				
$7 \cdot 10^9$	320,365				

its first $7 \cdot 10^9$, and for which FLA required at least 10^{10} attempts. These are the passwords to which FLA assigns low probabilities, despite being chosen by some users. Because PassGAN is able to model them, we conclude that the probabilities assigned by FLA to these passwords are incorrect. Figure 3 presents our result as the ratio between the passwords matched by FLA at a particular number of guessing attempts, and by PassGAN within its first $7 \cdot 10^9$ attempts. These results show that PassGAN is able to model a number of passwords more correctly than FLA. However, this advantage decreased as the number of attempts required to FLA to guess a password increased, i.e., as the estimated probability of that password decreased. This shows that, in general, the two tools agree on assigning probabilities to passwords.

C. *Evaluating PassGAN's Output Properties*

In this section, we take a deeper look at some of the properties of PassGAN's output. First, we evaluate PassGAN's ability to perform density estimation of the training dataset. We then provide insights on the size of its output space. We conclude with empirical observation on passwords generated by PassGAN that did not match our testing sets.

PassGAN's Probability Density Estimation. The ability of PassGAN to match passwords within a limited number of attempts depends on its ability to correctly estimate the frequency of passwords. A correct frequency estimate enables

TABLE IV: Matches generated by PassGAN and FLA in addition to the matches from HashCat Best64. The testing set used in this table was obtained by removing all passwords generated by HashCat Best64 from the LinkedIn testing set (33,394,178 passwords). Column (4) represent the number of matches obtained with the passwords generated by PassGAN minus the passwords generated by FLA (and vice versa, in Column (5)).

Unique Passwords	**(1) PassGAN**	**(2) FLA**	**(3) PassGAN ∪ FLA**	**(4) PassGAN, and not from FLA**	**(5) FLA, and not from PassGAN**
10^4	120	8	128	120	8
10^5	1,221	461	1,677	1,216	456
10^6	12,351	15,640	27,332	11,692	14,981
10^7	109,469	245,224	330,065	84,841	220,596
10^8	656,322	1,724,648	2,048,656	324,008	1,392,334
10^9	2,428,119				
$7 \cdot 10^9$	5,262,427				

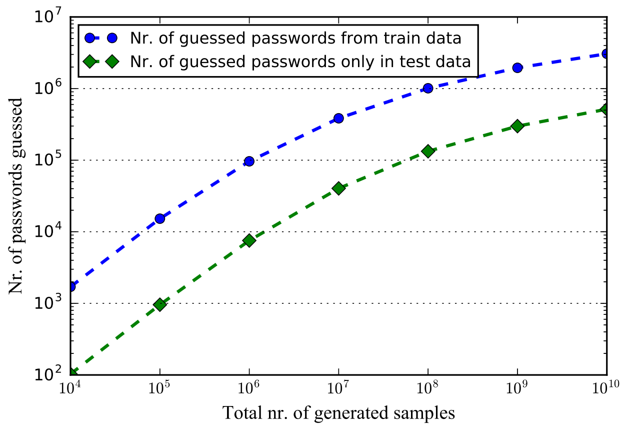

Fig. 4: Number of passwords generated by the GAN that appear in the training and in the testing set, as the number samples in the GAN's output increases.

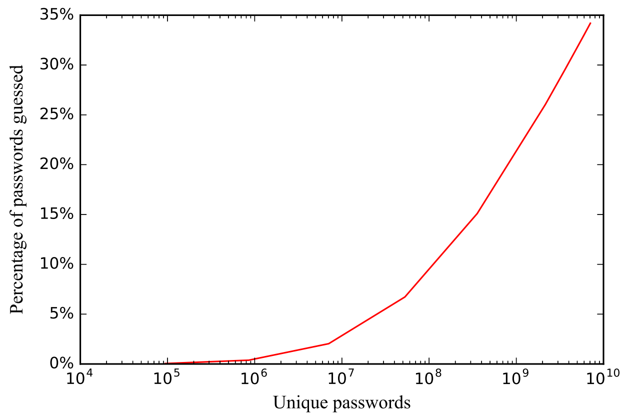

Fig. 5: Percentage of passwords generated by PassGAN that match passwords in the RockYou testing set. The x axis represents the number of unique samples produced by PassGAN.

PassGAN to match common passwords, and passwords with similar structure, within a small number of guesses. For instance, because `123456` represents more than 1% of the passwords in the training set, this password should also appear with roughly the same frequency in PassGAN's output.

To evaluate the ability of PassGAN to estimate the distribution of passwords from the training set, we generated a batch of 10^{10} passwords, and calculated the frequency of each password within the batch. We then compared these frequencies with the corresponding frequencies in the training set (ground truth). The results, summarized in Table V, show that PassGAN is able to correctly estimate the probabilities of many of the 50 most frequent passwords. Specifically, 40% of the 50 most frequent passwords generated by PassGAN are among the 100 most frequent passwords in the training set. Further, in our experiments PassGAN did not accurately model password probabilities for less likely passwords. Any improvement in the underlying GAN architecture that leads to better density estimation can potentially result in an improvement in the ability of PassGAN to correctly guess a large number of passwords within a small number of attempts.

Size of PassGAN's Output Space. To evaluate the size of the password space generated by PassGAN, we generated several passwords sets of sizes between 10^4 and 10^{10}. Our experiments show that, as the number of passwords increased, so did the number of unique (and therefore new) passwords. Results of this evaluation are reported in Table VI, and summarized in Figure 5. When we increased the number of passwords generated by PassGAN, the rate at which new unique passwords were generated decreased only slightly. Similarly, the rate of increase of the number of matches (shown in Table VI and Figure 4) diminished slightly as the number of passwords generated increased. This is to be expected, as the simpler passwords are matched early on, and the remaining (more complex) passwords require a substantially larger number of attempts in order to be matched.

A Closer Look at Non-matched Passwords. We inspected a list of passwords generated by PassGAN that did not match any of the testing sets and determined that many of these passwords are reasonable candidates for human-generated passwords. As such, we speculate that a possibly large number of passwords generated by the GAN, and that did not match our test sets, might still match user accounts from services other than RockYou and LinkedIn. We list a small sample of these passwords in Table VII.

TABLE V: Frequency of the 50 most common outputs of PassGAN, and corresponding frequency and rank in the RockYou training set. Passwords are sorted by the frequency in which they appear in PassGAN's output. "—" indicates that the password was not in the training set.

Password	Rank in Training Set	Frequency in Training Set	Frequency in PassGAN's Output
123456	1	0.9833%	1.0096%
123456789	3	0.25985%	0.222%
12345	2	0.26662%	0.2162%
iloveyou	5	0.16908%	0.1006%
1234567	7	0.07348%	0.0755%
angel	33	0.03558%	0.0638%
12345678	9	0.06983%	0.0508%
iloveu	21	0.04471%	0.0485%
angela	109	0.01921%	0.0338%
daniel	12	0.0521%	0.033%
sweety	90	0.02171%	0.0257%
angels	57	0.02787%	0.0245%
maria	210	0.01342%	0.0159%
loveyou	52	0.0287%	0.0154%
andrew	55	0.02815%	0.0131%
123256	301,429	0.00003%	0.013%
iluv!u	—	—	0.0127%
dangel	38,800	0.00018%	0.0123%
michel	1,442	0.00335%	0.0119%
marie	483	0.00755%	0.0118%
andres	223	0.01274%	0.0106%
lovely	15	0.0487%	0.0103%
123458	7,352	0.00076%	0.0099%
sweet	329	0.00999%	0.0097%
prince	243	0.01217%	0.0092%
ilove	2,177	0.00234%	0.0089%
hello	61	0.02648%	0.0086%
angel1	184	0.01459%	0.0085%
iluveu	58,131	0.00013%	0.0083%
723456	337,321	0.00003%	0.0082%
loveu	852	0.00505%	0.0082%
lovers	70	0.0253%	0.0082%
iluv!you	—	—	0.0082%
bella	732	0.00562%	0.0081%
andrea	43	0.03123%	0.0081%
iluveyou	183,386	0.00004%	0.0079%
kella	180,219	0.00004%	0.0076%
michelle	24	0.04312%	0.0074%
mariana	228	0.01265%	0.0074%
marian	681	0.00593%	0.0073%
daniela	95	0.02064%	0.0072%
dancer	122	0.01799%	0.0072%
lovery	46,470	0.00016%	0.0071%
dancel	42,692	0.00017%	0.007%
23456	3,976	0.00134%	0.007%
1g3456	—	—	0.007%
loveme	37	0.03302%	0.007%
jessie	213	0.01329%	0.0069%
buster	145	0.01619%	0.0068%
anger	172,425	0.00005%	0.0067%

TABLE VI: Number of passwords generated by PassGAN that match passwords in the RockYou testing set. Results are shown in terms of unique matches.

Passwords Generated	Unique Passwords	Passwords matched in testing set, and not in training set (1,978,367 unique samples)
10^4	9,738	103 (0.005%)
10^5	94,400	957 (0.048%)
10^6	855,972	7,543 (0.381%)
10^7	7,064,483	40,320 (2.038%)
10^8	52,815,412	133,061 (6.726%)
10^9	356,216,832	298,608 (15.094%)
10^{10}	2,152,819,961	515,079 (26.036%)
$2 \cdot 10^{10}$	3,617,982,306	584,466 (29.543%)
$3 \cdot 10^{10}$	4,877,585,915	625,245 (31.604%)
$4 \cdot 10^{10}$	6,015,716,395	653,978 (33.056%)
$5 \cdot 10^{10}$	7,069,285,569	676,439 (34.192%)

TABLE VII: Sample of passwords generated by PassGAN that did not match the testing sets.

love42743	ilovey2b93	paolo9630	italyit
sadgross	usa2598	s13trumpy	trumpart3
ttybaby5	dark1106	vamperiosa	˜dracula
saddracula	luvengland	albania.	bananabake
paleyoung	@crepess	emily1015	enemy20
goku476	coolarse18	iscoolin	serious003
nyc1234	thepotus12	greatrun	babybad528
santazone	apple8487	1loveyoung	bitchin706
toshibaod	tweet1997b	103tears	1holys01

V. Remarks

In this section, we summarize the findings from our experiments, and discuss their relevance in the context of password guessing.

Character-level GANs are well suited for generating password guesses. In our experiments, PassGAN was able to match 34.2% of the passwords in a testing set extracted from the RockYou password dataset, when trained on a different subset of RockYou. Further, we were able to match 21.9% of the password in the LinkedIn dataset when PassGAN was trained on the RockYou password set. This is remarkable, because PassGAN was able to achieve these results with no additional information on the passwords that are present only in the testing dataset. In other words, PassGAN was able to correctly guess a large number of passwords that it did not observe given access to nothing more than a set of samples.

Current rule-based password guessing is very efficient, but limited. In our experiments, rule-based systems were able to match or outperform other password guessing tools when the number of allowed guesses was small. This is a testament to the ability of skilled security experts to encode rules that generate correct matches with high probability. However, our experiments also confirmed that the main downside of rule-based password guessing is that rules can generate only a finite, relatively small set of passwords. In contrast, both PassGAN and FLA were able to eventually surpass the number of matches achieved using password generation rules.

As a result, the best password guessing strategy is to use multiple tools. In our experiments, each password guessing approach has an edge in a different setting. Our results confirm that combining multiple techniques leads to the best overall performance. For instance, by combining the output of PassGAN with the output of the Best64 rules, we were able to match 48% of the passwords in the RockYou testing dataset (which represents a 50.8% increase in the number of matches) and 30.6% of the passwords from the LinkedIn dataset—an increase of about 73.3%. Given the current performance of both PassGAN and FLA, it is not unlikely that tools alone will soon be able to replace rule-based password guessing tools entirely.

GANs are expressive enough to generate passwords from Markovian processes, rules, and to capture more general password structures. Our experiments show that PassGAN is competitive with FLA, which treats password guessing primarily as a Markovian process. Without any knowledge of password rules or guidance on password structure, PassGAN was able to match the performance of FLA within an order of magnitude of guesses by leveraging only knowledge that it was able to extract from a limited number of samples. Further, because GANs are more general tools than Markov models, in our experiment PassGAN was able to generate matching passwords that were ranked as very unlikely by FLA, using a limited number of guesses.

GANs generalize well to password datasets other than their training dataset. When we evaluated PassGAN on a dataset (LinkedIn [52]) distinct from its training set (RockYou [76]), the drop in matching rate was modest, especially compared to other tools. Moreover, when tested on LinkedIn, PassGAN was able to match the other tools within a lower or equal number of guesses compared to RockYou.

State-of-the-art GANs density estimation is correct only for a subset of the space they generate. Our experiments show that IWGAN's density estimation matches the training set for high-frequency passwords. This is important, because it allows PassGAN to generate highly-likely candidate passwords early. However, our experiments also show that as the frequency of a password decreases, the quality of PassGAN's density estimation deteriorates. While this becomes less relevant as PassGAN generates more passwords, it shows that the number of passwords that PassGAN needs to output to achieve a particular number of matches could greatly decrease if it is instantiated using a character-level GAN that performs more accurate density estimation. Similarly, a larger training dataset, coupled with a more complex neural network structure, could improve density estimation (and therefore PassGAN's performance) significantly.

VI. Conclusion

In this paper we introduced PassGAN, the first password guessing technique based on generative adversarial networks (GANs). PassGAN is designed to learn password distribution information from password leaks. As a result, and contrary to currents password guessing tools, PassGAN does not rely on any additional information, such as explicit rules, or assumption on the Markovian structure of user-chosen passwords. We believe that our approach to password guessing is revolutionary because, unlike current rule-based tools, PassGAN was able to generate passwords with no user intervention—thus requiring no domain knowledge on passwords, nor manual analysis of password database leaks.

We evaluated PassGAN's performance by testing how well it can guess passwords that it was not trained on, and how the distribution of PassGAN's output approximates the distribution of real password leaks. Our results show that PassGAN is competitive with state of the art password generation tools: in our experiments, PassGAN was always able to generate the same number of match as the other password guessing tools.

While PassGAN currently requires to output a larger number of passwords to compared to other tools to achieve the same number of matches, we believe that this cost is negligible when compared to the benefits of the proposed technique. Further, we believe that by training PassGAN on a larger dataset (which also allows us to deploy more complex neural network structures and more comprehensive training), the underlying GAN will perform more accurate density estimation, thus reducing the number of passwords needed to achieve a specific number of matches.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[2] A. Abdulkader, A. Lakshmiratan, and J. Zhang. (2016) Introducing deeptext: Facebook's text understanding engine. [Online]. Available: https://tinyurl.com/jj359dv

[3] (2018) Alphago – deepmind. [Online]. Available: https://deepmind.com/research/alphago/

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *CoRR*, vol. abs/1701.07875, 2017.

[5] "Arxiv summaries week 09/11 – passGAN: A deep learning approach for password guessing," *CS 294-131:Special Topics in Deep Learning*, 2017. [Online]. Available: https://berkeley-deep-learning.github.io/cs294-131-f17/arxiv.html

[6] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[7] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[8] H. Bidgoli, "Handbook of information security threats, vulnerabilities, prevention, detection, and management volume 3," 2006.

[9] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," *arXiv preprint arXiv:1607.04311*, 2016.

[11] ——, "Adversarial examples are not easily detected: Bypassing ten detection methods," *arXiv preprint arXiv:1705.07263*, 2017.

[12] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models." in *NDSS*, 2012.

[13] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[14] M. Cobb, "How machine learning-powered password guessing impacts security," *TechTarget*, 2017. [Online]. Available: http://searchsecurity.techtarget.com/tip/How-machine-learning-powered-password-guessing-impacts-security

[15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[16] J. Condliffe, "A pair of AIs have become very good at guessing your passwords," *Technology Review News*, 2017. [Online]. Available: https://www.technologyreview.com/the-download/608897/a-pair-of-ais-have-become-very-good-at-guessing-your-passwords/

[17] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer Berlin Heidelberg, 2013, pp. 403–410.

[18] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[19] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a? laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[20] M. Dirjish, "Artificial intelligence, slayer of passwords," *Sensors Online*, 2017. [Online]. Available: https://www.sensorsmag.com/embedded/artificial-intelligence-slayer-passwords

[21] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2634, 2015.

[22] B. Dorsey. (2017) Markov-chain password generator. [Online]. Available: https://github.com/brannondorsey/markov-passwords

[23] B. Duc, S. Fischer, and J. Bigun, "Face authentication with gabor information on deformable graphs," *IEEE Transactions on Image Processing*, vol. 8, no. 4, pp. 504–516, 1999.

[24] M. Dürmuth, F. Angelstorf, C. Castelluccia, D. Perito, and C. Abdelberi, "Omen: Faster password guessing using an ordered markov enumerator." in *ESSoS*. Springer, 2015, pp. 119–132.

[25] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber, "Using deep learning to enhance cancer diagnosis and classification," in *The 30th International Conference on Machine Learning (ICML 2013),WHEALTH workshop*, 2013.

[26] S. Fiegerman. (2017) Yahoo says 500 million accounts stolen. [Online]. Available: http://money.cnn.com/2016/09/22/technology/yahoo-data-breach/index.html

[27] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 136–148, 2013.

[28] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[29] J. W. Goerlich, "PassGAN for AI password guessing," *Stuck in Traffic*, 2017. [Online]. Available: https://www.youtube.com/watch?v=b92sTdRRwvs

[30] M. Golla. (2017) Password guessing using recurrent neural networks - the missing manual. [Online]. Available: https://www.password-guessing.org/blog/post/cupslab-neural-network-cracking-manual/

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[32] Google DeepMind. (2016) Alphago, the first computer program to ever beat a professional player at the game of GO. [Online]. Available: https://deepmind.com/alpha-go

[33] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[34] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[35] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017.

[36] HashCat. (2017). [Online]. Available: https://hashcat.net

[37] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, "LOGAN: Evaluating privacy leakage of generative models using generative adversarial networks," *CoRR*, vol. abs/1705.07663, 2017.

[38] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," *arXiv preprint arXiv:1706.04701*, 2017.

[39] K. J. Higgins, "The coolest hacks of 2017," *Dark Reading*, 2017. [Online]. Available: https://www.darkreading.com/threat-intelligence/the-coolest-hacks-of-2017/d/d-id/1330699

[40] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," *In ACM CCS'17*, 2017.

[41] B. Hitaj, P. Gasti, G. Ateniese, and F. Pérez-Cruz, "Passgan: A deep learning approach for password guessing," *CoRR*, vol. abs/1709.00440v1, 2017. [Online]. Available: http://arxiv.org/abs/1709.00440v1

[42] M. Hutson, "Artificial intelligence just made guessing your password a whole lot easier," *Science*, 2017. [Online]. Available: http://www.sciencemag.org/news/2017/09/artificial-intelligence-just-made-guessing-your-password-whole-lot-easier

[43] E. Intini, "Lìntelligenza artificiale indovina le password," *Focus IT*, 2017. [Online]. Available: https://www.focus.it/tecnologia/digital-life/lintelligenza-artificiale-indovina-le-password

[44] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[47] C. Lab. (2016) Fast, lean, and accurate: Modeling password guessability using neural networks (source code). [Online]. Available: https://github.com/cupslab/neural_network_cracking

[48] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," *arXiv preprint arXiv:1509.01549*, 2015.

[49] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems 2, NIPS 1989*. Morgan Kaufmann Publishers, 1990, pp. 396–404.

[50] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[51] Y. LeCun, K. Kavukcuoglu, C. Farabet *et al.*, "Convolutional networks and applications in vision." in *ISCAS*, 2010, pp. 253–256.

[52] LinkedIn. Linkedin. [Online]. Available: https://hashes.org/public.php

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

[54] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 689–704.

[55] I. Madan, "Up to speed on AI & deep learning: October update," *Hackernoon*, 2017. [Online]. Available: https://hackernoon.com/up-to-speed-on-deep-learning-october-update-815f5eef0e2b

[56] A. Manning, "Researchers show how a.i. is the end of passwords as we know them," *Inverse*, 2017. [Online]. Available: https://www.inverse.com/article/36604-ai-cracking-passwords

[57] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 2016.

[58] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 175–191. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher

[59] C. Metz. (2016) Google's GO victory is just a glimpse of how powerful ai will be. [Online]. Available: https://tinyurl.com/l6ddhg9

[60] M. Mimoso, "Deep-learning passGAN tool improves password guessing," *Threatpost*, 2017. [Online]. Available: https://threatpost.com/deep-learning-passgan-tool-improves-password-guessing/128039/

[61] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[63] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979.

[64] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proceedings of the 12th ACM conference on Computer and communications security*. ACM, 2005, pp. 364–372.

[65] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, "Jointly modeling embedding and translation to bridge video and language," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4594–4602, 2016.

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

[67] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, 2015.

[68] C. Percival and S. Josefsson, "The scrypt password-based key derivation function," Tech. Rep., 2016.

[69] S. Perez. (2017) Google plans to bring password-free logins to android apps by year-end. [Online]. Available: https://techcrunch.com/2016/05/23/google-plans-to-bring-password-free-logins-to-android-apps-by-year-end/

[70] H. P. position Markov Chains. (2017). [Online]. Available: https://www.trustwave.com/Resources/SpiderLabs-Blog/Hashcat-Per-Position-Markov-Chains/

[71] "Powerful password crackers may be closer than you think, say stevens institute of technology researchers," *Stevens Institute of Technology*, 2017. [Online]. Available: https://tinyurl.com/y76zejnx

[72] T. P. Project. (2017). [Online]. Available: http://thepasswordproject.com/leaked_password_lists_and_dictionaries

[73] N. Provos and D. Mazieres, "Bcrypt algorithm," in *USENIX*, 1999.

[74] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations*, 2016.

[75] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[76] RockYou. (2010) Rockyou. [Online]. Available: http://downloads.skullsecurity.org/passwords/rockyou.txt.bz2

[77] H. Rules. (2017). [Online]. Available: https://github.com/hashcat/hashcat/tree/master/rules

[78] J. T. R. K. Rules. (2017). [Online]. Available: http://contest-2010.korelogic.com/rules.html

[79] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[80] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[81] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 3–18.

[82] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "Hmog: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 877–892, 2016.

[83] T. SPIDERLABS. (2012) Korelogic-rules. [Online]. Available: https://github.com/SpiderLabs/KoreLogic-Rules

[84] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[85] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1701–1708. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.220

[86] J. the Ripper. (2017). [Online]. Available: http://www.openwall.com/john/

[87] J. the Ripper Markov Generator. (2017). [Online]. Available: http://openwall.info/wiki/john/markov

[88] I. Thomson, "AI slurps, learns millions of passwords to work out which ones you may use next," *The Register*, 2017. [Online]. Available: https://www.theregister.co.uk/2017/09/20/researchers_train_ai_bots_to_crack_passwords/

[89] I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf, "Adagan: Boosting generative models," *arXiv preprint arXiv:1701.02386*, 2017.

[90] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX*, 2016.

[91] J. Vijayan, "PassGAN: Password cracking using machine learning," *Dark Reading*, 2017. [Online]. Available: https://www.darkreading.com/analytics/passgan-password-cracking-using-machine-learning/d/d-id/1329964

[92] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 391–405.

[93] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse, "On the quantitative analysis of decoder-based generative models," *arXiv preprint arXiv:1611.04273*, 2016.

[94] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[95] X. Zhang and Y. A. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710v5*, 2016.

[96] Y. Zhong, Y. Deng, and A. K. Jain, "Keystroke dynamics for user authentication," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 117–123.

Appendix A
Configuration Parameters for Running FLA

We run the code that implements the password metering and guessing tool introduced in [58] using the parameters listed in Table VIII.

TABLE VIII: Training configuration used for FLA

Configuration Parameters	Value
training_chunk	128
training_main_memory_chunk	23679744
min_len	4
max_len	10
context_length	10
chunk_print_interval	1000
layers	2
hidden_size	1000
generations	20
training_accuracy_threshold	-1
train_test_ratio	20
model_type	JZS2
train_backwards	True
dense_layers	1
dense_hidden_size	512
secondary_training	False
simulated_frequency_optimization	False
randomize_training_order	True
uppercase_character_optimization	False
rare_character_optimization	False
rare_character_optimization_guessing	False
no_end_word_cache	True
intermediate_fname	data.sqlite
save_model_versioned	True