PassGAN for Honeywords: Evaluating the Defender and the Attacker Strategies

Muhammad Ali Fauzi[1], Bian Yang[1], and Edlira Martiri[1]

[1] Norwegian University of Science and Technology, Gjøvik, Norway
{muhammad.a.fauzi, bian.yang, edlira.martiri2}@ntnu.no

Abstract. The main challenge in a honeywords system is how to generate artificial passwords (honeywords) that are indistinguishable from the genuine password (sugarword). It is straightforward to consider the PassGAN for generating honeywords from the defender's perspective. In this work, we analyze a game situation between the defender and the attacker assuming the two parties exploit the PassGAN for their own competing advantage, i.e., the defender uses the generator model of PassGAN to generate honeywords and the attacker uses the discriminator model of PassGAN to detect the sugarword. In this game, we investigate the feasibility of PassGAN as a honeywords generation strategy and the possible strategies that can be used by the defender and the attacker to reach their goal. The best strategy for the attacker is to use a large number of iterations and to use the same dataset as the defender. From the defender's point of view, the strategy of using many iterations is also beneficial to reduce the attacker's success rate.

Keywords: Honeywords, PassGAN, Attacker, Defender, Strategy.

1 Introduction

A password remains the most widely used identity authentication method due to its simplicity and familiarity to users and developers [21]. Unfortunately, there have been many password data leaks recently including data from well-known organizations such as Rock-you [20], Dropbox [16], Yahoo [15], etc. Even worse, the leaks show that most users prefer poor passwords for their accounts that enable attackers to guess them easily using cracking techniques such as dictionary attack, despite that the passwords are saved in the hashed form [9, 11]. Most of these breaches were only discovered months or even years after they first happened. During the period between the breach and its discovery, the attacker surely had exploited most of these passwords, some had even published or sold them online. Therefore, it is important to have not only a mechanism to improve security but also a mechanism to detect password data leaks quickly [10].

Some approaches have been proposed to handle a password data breach [4, 19]. A promising one is introduced by Juels and Rivest called honeywords [18]. In this system, the mixture of both real passwords (sugarwords or sugars) and decoy passwords (honeywords or honeys) are stored in the password file. If an attacker manages to steal the file containing the password and successfully resolves all hash values in the file, she or

he still has to be able to distinguish the real password from the artificial ones. When an attacker tries to log in using a honeyword, the system will detect it and provide an alarm to denote that there is an attack on the password file. This technique is more practical because it only needs a few changes on the system, on both the server and client sides [12].

The main challenge for the system administrator in this approach is how to generate honeywords that are difficult to distinguish from the real ones. Some previous methods for honeyword generation include random-replacement [18], utilizing existing passwords [8], questionnaire-based [7], text and image-based non-realistic honeyword generation [3], and paired distance protocol (PDP) [6]. One of the newest methods that can be that shows potential is PassGAN [17].

PassGAN is designed to be a password guessing method [17] that utilizes a Generative Adversarial Network (GAN) [13] to learn from the real passwords as training data and generate password guesses that have a high level of similarity to the real passwords. PassGAN does not require prerequisite knowledge or intuition from experts about what type of passwords are often chosen by users since it will autonomously learn from the real passwords. Besides, PassGAN for honeywords generation is classified as a legacy-UI method because it does not require user intervention. This kind of method is preferred due to usability advantage [18]. Therefore, PassGAN can be a good tool for honeyword generation from the defender's perspective.

On the other hand, attackers are also becoming smarter and are trying to find ways to pick the right password. The attackers could use various password guessing tools and machine-learning techniques to distinguish the real passwords from the decoys. The worst case scenario for the defender is when the attacker also uses PassGAN to determine the correct password. This condition raises a question of how feasible PassGAN is as a honeyword generation strategy and what are the best strategies for both the defender and the attacker in this situation.

In this study, we assume that the attacker has managed to crack the hashed password file that is leaked from the defender and could use PassGAN to distinguish the sugarword from the honeywords. We will analyze the feasibility of PassGAN for the generation of honeywords in this case. We will also investigate strategies that can be used by the attacker to make this distinguishing process more accurate and strategies that can be used by the defender (e.g., the system administrator) to better secure their honeywords system against this kind of attack.

Most previous researches on honeywords (e.g. [8, 7, 6, 18]) only evaluate the honeywords generation strategies heuristically. In this study, we will conduct an evaluation of PassGAN for honeywords generation empirically with some near real-world scenarios and use datasets from some real systems (e.g. leaked password data).

2 Background and Related Works

2.1 Honeywords

Honeywords is a promising security mechanism introduced by Juels and Rivet [18]. The principal idea is to mix each real password with a number of fake passwords in order to confuse attackers. The real password is called a sugarword while the decoys are called honeywords. The combination of the two is often referred to as sweetwords. When the attacker succeeds in cracking passwords from the leaked password file, she or he still has to be able to choose which password is the genuine one out of all the cracked sweetwords. Once the attacker logs in using a honeyword, an alarm will be triggered to notify the defender that there is a malicious password attempt to the system and high confidence that the corresponding identity principal's shadow password file was leaked from the system and cracked.

Generally, the system works as follows. During the registration process, the user will choose a username and a password. Following that, *k-1* honeywords associated with the password will be generated so that it has *k* sweetwords. The sweetwords, together with the username and the other information about the user, are then stored in the login server after which the order of the passwords is shuffled. Subsequently, the information about the user index and the index of the real password will be saved into a checker-sever called a honeychecker. Since the honeychecker only has information about user index and real password index, the honeychecker can be used only if we have access to user information in the login server. In this system, it is assumed that the attacker does not manage to steal both the login server and the honeychecker data in the same period.

During the login phase, the user enters his username and password. The system then looks for the matching record in the login server. If the username exists, after performing the hashing calculation to the entered password, the system then checks whether the password is in the sweetwords stored for the corresponding user. If the password is not found in the sweetwords, then the password entered is incorrect and login is denied. Otherwise, if the password is found, the system sends information about the user index and index of the entered password into the honeychecker. Honeychecker then makes a comparison between the entered password index and the real password index for the corresponding user in the database. If the two have the same index, then the login is successful. Alternatively, the honeychecker sends an alert to the system administrator about a password data breach or conducts other procedures in accordance with the policy that has been previously determined.

2.2 PassGAN

PassGAN is a deep learning-based password guessing method developed by Hitaj et al. [17]. PassGAN uses a Generative Adversarial Network (GAN) to autonomously learn from the real password dataset and then generate passwords that have the similar distribution to the dataset. GAN is a deep learning approach for estimating generative models through an adversarial process invented by Goodfellow et al. [13].

GAN is comprised of two neural networks that are pitted against each other: a generative model *G* and a discriminative model *D*. *G* studies the distribution of training data and creates new data instances or samples with similar distributions, while *D* computes the probabilities whether each sample is from the real training data or generated by *G*. *G* and *D* are trained simultaneously where the goal of *G* is to maximize the probability of *D* making a mistake while in contrast, *D* aims to successfully detect the fake samples. Competition in this game drives both *G* and *D* to improve their methods until the generated samples cannot be distinguished from the real data. Recently, there have been many attempts to improve GAN. One of the most promising ones is IWGAN [14] due to its ability to make training more stable. IWGAN has also been successfully implemented for text generation and achieves improvements in performance.

In PassGAN, a generator model is trained to learn about the characteristics and structures of passwords from training data (e.g. leaked password data). The generative model then generates some fake passwords based on the training model. At the same time, the discriminator model is simultaneously trained to distinguish the real password from the fake ones.

3 Our Work

3.1 Game Situation

In this work, we simulate a game between the defender and the attacker. As seen in Figure 1, three datasets are used for the game: the defender's dataset for her or his PassGAN training, the attacker's dataset for her or his PassGAN training, and the system dataset as the real passwords (sugarwords). The game flowchart can be divided into two parts: the defender's work and the attacker's work.

The defender and the attacker train their own PassGAN model separately using their own dataset. The defender uses *p* iterations to train her or his PassGAN while the attacker uses *q* iterations to train her or his PassGAN. Both the attacker and the defender can use any number of iterations for their training process as it is one strategy that can be exploited by each party. After the training is complete, the defender takes the generator model of her or his PassGAN, which is originally comprised of generator and discriminator model, and then uses it to generate $k-1$ honeywords for each account's sugarword in the system dataset. These generated honeywords are then combined with each sugarword from the system dataset to compose *k* sweetwords for each account.

Meanwhile, the attacker takes the discriminator model of her or his PassGAN and then uses it to determine the sugarword among the sweetwords for each account. The guessing process works as follows. For each account, the attacker uses the discriminator model to compute the probability of each sweetword becoming a sugarword. Then, the sweetwords are sorted in descending order based on their probability value. The sweetword with the highest probability is then considered to be the real password of the account and will be submitted to the system by the attacker.

The guessed passwords are then evaluated. The attacker is considered successful if she or he can guess the correct password for each account. Otherwise, the attacker is

considered to have failed. The success rate of the attacker will be used to evaluate both the attacker's and defender's strategies.

The objective of the defender in this game is to generate indistinguishable honeywords that can make the attacker's success rate low. On the contrary, the attacker aims to achieve the best discriminator performance thereby maximizing the probability of guessing a correct password and attain a high success rate. In this game situation, we analyze the best strategy the two parties can exploit for their own competing advantage.

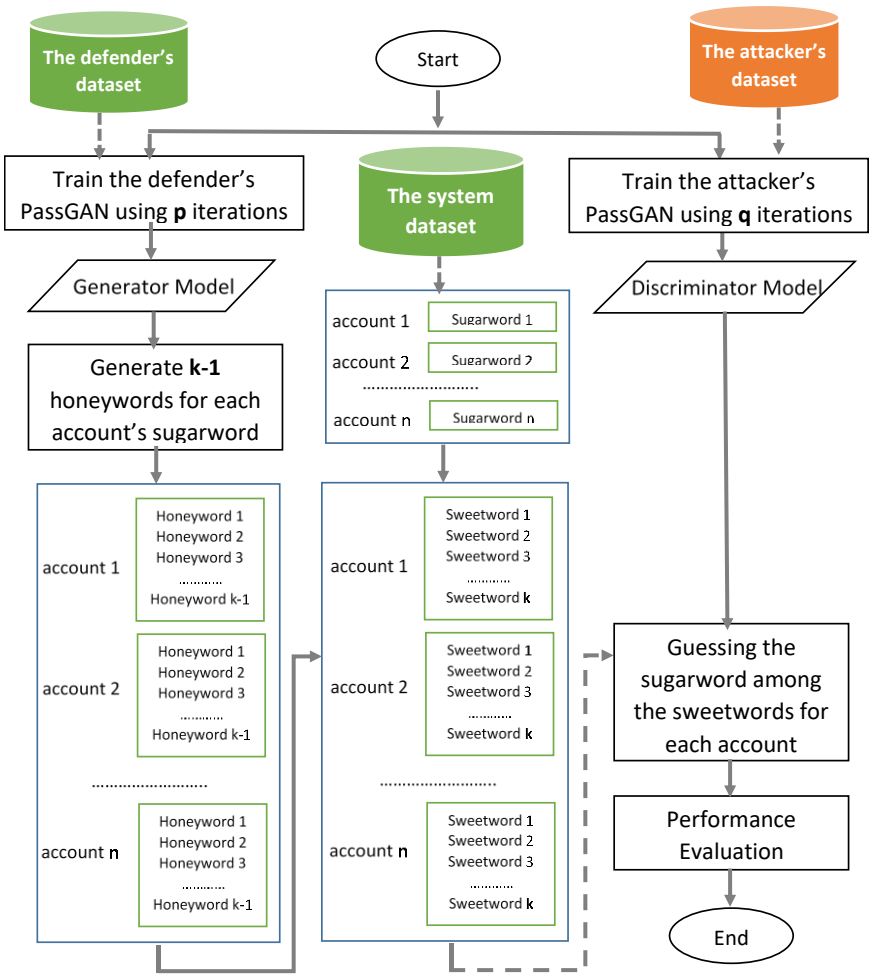


Fig. 1. The attacker and the defender game flow.

3.2 Dataset

Datasets used for this work are leaked passwords from Rock-you [5], Dropbox [1], and Linkedin [2]. The RockYou, Dropbox, and Linkedin datasets contain 21,315,673,

7,884,855, and 10,000 passwords respectively. These three password datasets are quite similar because they were leaked from online media platforms that had similar users, from various professions who were mostly aged 18-34, and the password profile in these three datasets is very similar, e.g., there is no need for special characters or a combination of upper-case and lower-case letters.

3.3 Experimental Setup

In this experiment, we have made some assumptions as follows:

- The defender uses the generator model of PassGAN to generate honeywords.
- The defender uses a public password dataset (e.g. leaked passwords) as her or his training data.
- The attacker manages to steal the data and crack all the hashed passwords.
- The attacker uses the discriminator model of PassGAN to guess the sugarword.
- In the real world, the attacker most likely does not know and does not have access to the defender's training dataset. Therefore, the attacker is likely to use a different dataset from the defender's dataset for the PassGAN training. However, we also conduct an experiment when the attacker and the defender use the same dataset to simulate a worst case scenario for the defender.

The dataset used by the defender in this work is the RockYou (RY) dataset while the attacker uses the Dropbox (Db) dataset. In a scenario when the attacker and the defender use the same dataset, the RY dataset is used for the experiment. The Linkedin (LI) dataset containing 10,000 accounts' real passwords is used as the system dataset's sugarwords (n = 10000). For each account's sugarword, 9 honeywords are generated and then these are combined to compose 10 sweetwords (k = 10).

The following is the detail of the two scenarios:

- Scenario 1: The defender and the attacker use the same number of iterations (p = q). A various number of iterations are used for the experiment (p = q = {5000, 10000, ..., 195000}). In this scenario, two dataset variations (both parties use a different dataset and both parties use the same dataset) are also examined.
- Scenario 2: The attacker and the defender use a different number of iterations. The attacker uses a fixed number of iterations (p = 100000), while the defender uses several numbers of iterations (q = {5000, 10000, ..., 195000}). In this scenario, two dataset variations (both parties use a different dataset and both parties use the same dataset) are also examined.

3.4 Performance Evaluation

Juels and Rivest [18] proposed flatness measurement to evaluate honeywords generation strategies. A honeywords generation method is called ϵ-flat when the attacker has a maximum success rate ϵ given a one-time opportunity to choose a correct pass-

word. A generation strategy is called "perfectly flat" if the attacker's maximum success rate $\epsilon=1/k$. If the success rate ϵ is not much greater than $1/k$, it is called "approximately flat". Therefore, the goal of both the defender and the attacker in this experiment is measured using the attacker's success rate. The formula is as follows:

$$\epsilon = \frac{NoC}{NoA} \tag{1}$$

where ϵ is the attacker's success rate, *NoC* is the number of accounts whose passwords have been guessed correctly and *NoA* is the total number of accounts. In this experiment, since the number of sugarwords used is 10,000, then the number of accounts (*NoA*) is always 10,000. Besides, we will also calculate the attacker's success rate when the attacker is given the opportunity more than once to choose the correct password in case there are honeywords systems that allow more than one guess.

4 Experimental Results

4.1 Scenario 1

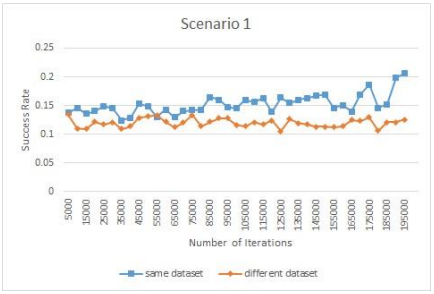
The experiment results from scenario 1 are shown in Figure 2. Generally, the attacker's success rate in scenario 1 tends to be very low. This means that the generated honeywords are hard to distinguish from the sweetwords so that the attacker discriminator model can only correctly guess a few passwords. Based on Figure 2a, as expected, the attacker's success rate given only one guess when both parties use the same dataset is higher by almost 0.1 than when they use a different dataset. The use of the same dataset makes the attacker's discriminator model learn from the same defender's dataset so that it can more successfully distinguish the honeywords. When the attacker and the defender use a different dataset, the success rate is relatively stable even though the number of iterations increases. The success rate only fluctuates by a very small margin, between 0.1 and 0.15. Meanwhile, when the attacker and the defender use the same dataset, the success rate increases as more iterations are used. The highest success rate (0.21) is obtained when both parties use 195000 iterations.

Based on Figure 2b, the success rate when both parties use the same dataset is also higher than when they use different datasets. However, the difference is relatively small and the success rates in these two conditions are not much greater than the "perfectly flat" method. Therefore, this honeywords generation method can be considered as "approximately flat".

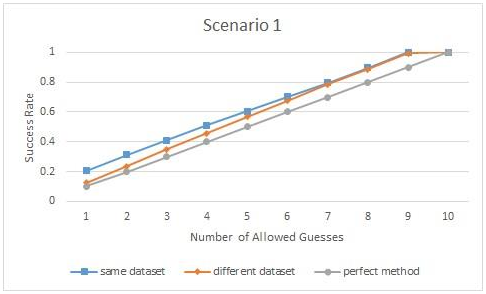
4.2 Scenario 2

The experiment results from scenario 2 are shown in Figure 3 and Figure 4. Generally, the use of a different number of iterations does not show significant differences as the attacker's success rate in scenario 2 also tends to be very low. Based on Figure 3, the use of the same dataset also gives a higher success rate for the attacker than the use of a different dataset. In the same dataset condition, the success rate decreases, slightly, as

the number of defender's iterations increases and becomes larger than the attacker's number of iterations. Meanwhile, when the attacker uses a different dataset from the defender's dataset, the success rate is relatively stable as it only fluctuated by a very small margin.



(a) The attacker's success rate given only 1 opportunity to guess the correct password in scenario 1. In this figure, the attacker and the defender use several iteration variations.



(b) The attacker's success rate given more than 1 opportunity to guess the correct password in scenario 1. In this figure, the defender and the attacker use 195000 iterations.

Fig. 2. The scenario 1 experiment results

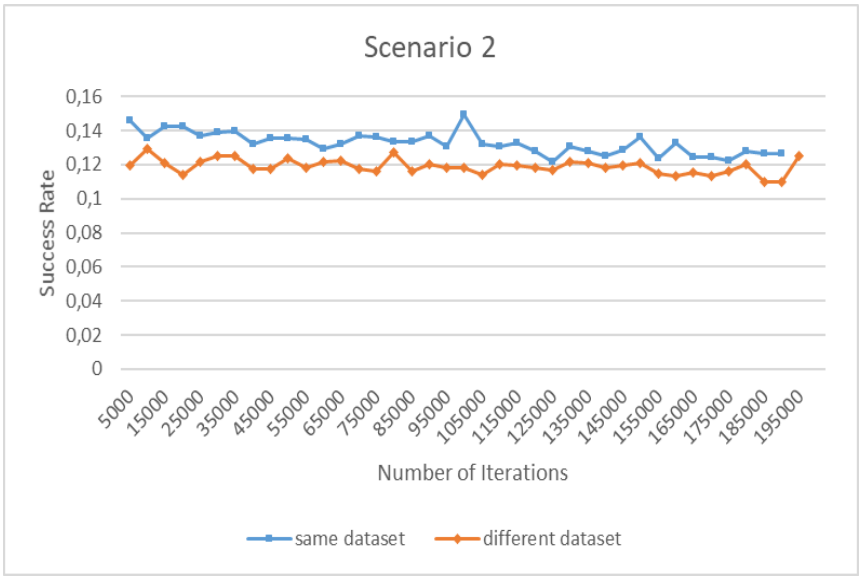
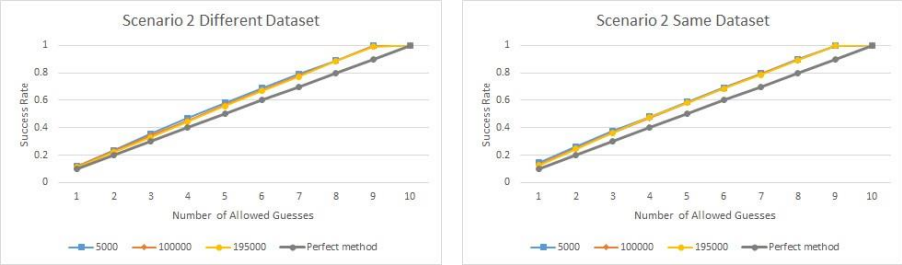


Fig. 3. The attacker's success rate given only 1 opportunity to guess the correct password in scenario 2. In this figure, the attacker uses a fixed number of iterations (100000), while the defender uses several numbers of iterations.



(a) The attacker and the defender use different datasets.

(b) The attacker and the defender use the same dataset.

Fig. 4. The attacker's success rate given more than 1 opportunity to guess the correct password in scenario 2. In these figures, the attacker uses a fixed number of iterations (100000), while the defender uses several numbers of iterations.

Based on Figure 4a and Figure 4b, the use of a different number of iterations by the defender does not show significant differences in success rate. The highest success rate is obtained when the defender uses 5000 iterations (fewer than the attacker's number of iterations) and the lowest success rate is obtained when the defender uses 195000 iterations (more than the attacker's number of iterations). However, the difference is barely noticeable because it is very small.

5 Conclusion

In this work, we analyze the feasibility of PassGAN for a honeywords generation method. Furthermore, we also investigate some possible strategies that can be used by the attacker and the defender in a PassGAN based honeywords system. The defender uses the generator model of PassGAN to generate high-quality fake passwords while it is assumed that the attacker can manage to steal and crack all the hashed password data and then uses the discriminator model of PassGAN to distinguish the real passwords from the fake ones. Based on the experiment results, PassGAN is feasible for use as a honeywords generation strategy. PassGAN is "approximately flat" even when the attacker also uses PassGAN to distinguish the sugarword from the honeywords.

The best strategy that the attacker can employ is to use the same dataset as the defender's dataset. In addition. the attacker can also use a large number of iterations as their strategy. A greater number of iterations is proven to enable the attacker to increase their success rate even though this increase is very small. From the defender's perspective, the use of many iterations is also beneficial to enable the defender to reduce the attacker's success rate.

In future works, several other strategies for both the attacker and the defender can also be examined. Despite its ability to generate high quality artificial passwords, PassGAN is not a user context-aware method. If a user uses a password that is correlated to her or his data (e.g. birthday, pet name, etc.), the PassGAN-based generated honeywords would be easily distinguished. Therefore, the use of a targeted guessing

attack to examine the PassGAN-based honeywords generation strategy would be relevant for future works.

References

1. Leaks dropbox, https://hashes.org/leaks.php?id=91.
2. Leaks linkedin, https://hashes.org/leaks.php?id=68.
3. Akshaya, K., Dhanabal, S.: Achieving flatness from non-realistic honeywords. In: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). pp. 1–3. IEEE (2017).
4. Bojinov, H., Bursztein, E., Boyen, X., Boneh, D.: Kamouflage: Loss-resistant password management. In: European symposium on research in computer security. pp. 286–302. Springer (2010).
5. Brannondorsey: Passsgan (2018), https://github.com/brannondorsey/PassGAN/releases/download/data/rockyou-train.txt.
6. Chakraborty, N., Mondal, S.: On designing a modified-ui based honeyword generation approach for overcoming the existing limitations. Computers & Security **66**, 155–168 (2017).
7. Chakraborty, N., Singh, S., Mondal, S.: On designing a questionnaire based honeyword generation approach for achieving flatness. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 444–455. IEEE (2018).
8. Erguler, I.: Achieving flatness: Selecting the honeywords from existing user passwords. IEEE Transactions on Dependable and Secure Computing **13**(2), 284–295 (2015).
9. Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th international conference on World Wide Web. pp. 657–666. ACM (2007).
10. Florˆencio, D., Herley, C., Van Oorschot, P.C.: An administrator's guide to internet password research. In: 28th Large Installation System Administration Conference (LISA14). pp. 44–61 (2014).
11. Furnell, S.M., Dowland, P., Illingworth, H., Reynolds, P.L.: Authentication andsupervision: A survey of user attitudes. Computers & Security **19**(6), 529–539 (2000).
12. Genc, Z.A., Karda¸s, S., Kiraz, M.S.: Examination of a new defense mechanism: Honeywords. In: IFIP International Conference on Information Security Theory and Practice. pp. 130–139. Springer (2017).
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair,S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014).
14. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in neural information processing systems. pp. 5767–5777 (2017).
15. Hackett, R.: Yahoo raises breach estimate to full 3 billion accounts, by farbiggest known (Oct 2017), https://blog.dropbox.com/topics/company/resettingpasswords-to-keep-your-files-safe.
16. Heim, P.: Resettingpasswords to keep your files safe (Aug 2016), https://blog.dropbox.com/topics/company/resetting-passwords-to-keep-yourfiles-safe.
17. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: Passgan: A deep learning approach for password guessing. In: International Conference on Applied Cryptography and Network Security. pp. 217–237. Springer (2019).

18. Juels, A., Rivest, R.L.: Honeywords: Making password-cracking detectable. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 145–160. ACM (2013).
19. Rao, S.: Data and system security with failwords (Jul 20 2006), US Patent App.11/039,577.
20. Siegler, M.: One of the 32 million with a rockyou account? you may want to change all your passwords. like now (Dec 2009), https://techcrunch.com/2009/12/14/rockyou-hacked/.
21. Wang, D., Cheng, H., Wang, P., Yan, J., Huang, X.: A security analysis of honeywords. In: NDSS (2018).