ABSTRACT

Generative adversarial networks are a class of machine learning framework which is used to generate new data based on the data which it is trained on. It is unsupervised, therefore it identifies and learns if any patterns present in data. It mainly consists of two components namely generator and discriminator. Discriminator tries to differentiate actual data and generated data and the generator generates new data. They compete with each other and due to this generator will be able to generate useful data.

In this project, we use this model to generate passwords. There are many existing techniques to generate password guesses which include brute force, rule based generation, etc. Creating rules requires expertise and is time consuming. This model will be able to learn hidden patterns automatically and thus is easier to use. In addition, this can also be used to determine the strength of the password.

INTRODUCTION

Passwords are being used for authentication purposes for a long time. It is easy and convenient for implementing and using it. Companies need to make sure that the user's passwords are not compromised. If compromised it can be misused which affects the user's account which may lead to loss of money, comprise identity, etc.

In the past there were many attempts to hack the databases and also has also happened which is concerning. The passwords are stored in hashed form in the database. If the passwords are strong enough, it will be very difficult to hack it. There are numerous ways like dictionary attack, rule based attack, etc to hack the passwords. These methods require expertise and are time consuming tasks. The output space is also limited.

Generative adversarial networks can be used to generate passwords that humans are likely to use. We need to train the model based on a list of real passwords that people have used. This data is available as there instances in which the compromised passwords were put online.

These GAN's without the domain knowledge will be able to understand how passwords are created and once properly trained, they will generate passwords which are likely used. This approach does not require for us to input any rules, only a list of passwords are required to train the model.

Well trained models can be used to substitute or used in addition to the existing tools for password cracking which may lead to better results. Certain patterns in passwords which might not be easy to identify by humans can be understood by this model. The output space of this model is not limited and can be used to generate many passwords. In addition to its application for password cracking, it can be used to determine the strength of any password so that users make sure their password is strong is less likely to be guessed.

The model consists of two main components called generator and discriminator. Discriminator outputs the likeness of the password being created by humans. Generator tries to generate more human-like passwords and fool the discriminator.

PROBLEM DEFINITION

Password generation is a very common security issue. Many users face the problem of choosing good passwords and try to make sure it is strong and not guessable.

Train a generative adversarial network which has 2 components namely generator and discrimintor. Generator should generate password guesses to break a password scheme and the discriminator tries to prevent the generator from breaking the password scheme.

In other words, discriminator tries to distinguish between real password and the generated password and the generator tries to generate a real looking password to fool discriminator.

There should be provision to train the model based on different datasets and provision to generate the required number of passwords. There should also be provision to determine the strength of a given password based on likeness of being able to guess the password.

The proposed model can be used as a security measure where the user can check for easily crackable passwords. It can find its applications in checking password strength.

Compare this model with other tools like hashcat and check the strength of the password using zxcvbn.

LITERATURE SURVEY

In this chapter, we analyse the research made and results obtained in the field of the project, by taking into account the various parameters and extent of the project.

3.1 Adversarial Password Cracking

In this paper, a generative adversarial network is trained with a generator trying to generate guesses to break a password scheme. The author has later compared the strength of this adversarial model with rule based methods such as hashcat . Later, the passwords cracked by PassGAN are compared with results of zxcvbn, to determine if the passwords cracked by PassGAN were actually hard to crack.

The author suggests using a machine learning based approach for password guessing which is called PassGAN. PassGAN is a generative adversarial network where the generator tries to generate a fake password to fool the discriminator and discriminator tries to distinguish real passwords from fake passwords (passwords generated by generator). They both end up playing a minimax game and optimize the value function given by Equation 1.

(1) $\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log(D(x))] + E_{z \sim p_z(z)}[\log(1 - G(z))]$

The author has trained the PassGAN model on two datasets - one is RockYou and second is AshleyMadison dataset. Further, an extended version of RockYou dataset was created by including slightly modified variations of the passwords, to represent a more realistic set of passwords.
The network was trained upto 70k iterations and a plot of training loss versus iterations is represented by Figure 1.

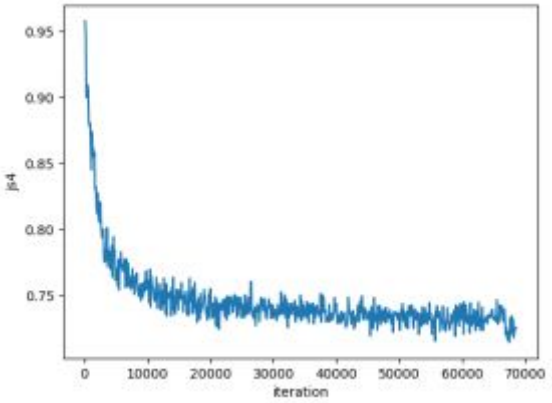

Figure 1 : Training loss v/s number of iterations (from paper [1])

After training, the set of passwords cracked using PassGAN was evaluated using zxcvbn and the results for RockYou and extended RockYou dataset are summarized in Table 1. From this table, we can infer that the average number of guesses for extended RockYou is high since it is a much randomized version of rockYou dataset.

Finally, the author concludes by mentioning that PassGAN can also be used to determine the strength of the password by counting the number of trials it takes before arriving at the correct password.

Data	Avg. Score	Avg. Guesses	Log Guesses
RockYou	1.5925	5.32×10^6	6.2561
Ext RockYou	1.5561	6.8×10^7	6.166

Table 1 : Average guesses for RockYou and extended RockYou datasets (from paper [1])

3.2 Presenting New Dangers: A Deep Learning Approach to Password Cracking

In this paper, the author argues that GANs generate high quality password guesses, and can match 51 to 73 percent more passwords than the traditional rule based methods. This shows that deep learning methods can be used for password cracking.
The paper begins by stating how carelessness of users can be exploited by malicious hackers in this online world.
Next, the paper presents us with various kinds of attacks used by state of the art password guessing tools such as hashcat. Brute force attack generates passwords by going through a combination of characters upto a given length. Dictionary based attacks use a list of words, either from a dictionary or commonly used passwords revealed from past password leaks, and then turn them into hashes. Then, it checks these hashes against the unknown hash to crack the unknown hash. Dictionary attacks often succeed because people have a natural tendency to choose ordinary words or common passwords with little variants. Rule based attacks use a set of rules on top of the dictionary list. For example, there could be a rule for concatenating all passwords with some numbers since these kinds of passwords are highly probable.
The paper evaluated the performance of PassGAN by running the model on different datasets. The results prove that PassGAN can match 51 to 73 percent more passwords than the traditional rule based methods. Additionally, it was found that the best overall performance was achieved when PassGAN was combined with other techniques, specifically when PassGAN and HashCat were used jointly. This concludes that PassGAN was able to generate real-like passwords and when hashcat was used on these passwords, it was able to perform much better since the password list generated by PassGAN consisted of highly probable passwords.

The author concludes by stating that PassGAN can be effectively used to generate passwords that resemble real passwords since it can extract properties of passwords which could be indistinguishable to human observation. Also, PassGAN can be used without a priori knowledge of the structure of passwords. Also, PassGAN can generate passwords indefinitely whereas other tools can generate a limited number of passwords. The best overall performance was achieved when PassGAN was combined with other rule based techniques.

3.3 PassGAN: A Deep Learning Approach for Password Guessing

This paper is the first attempt to apply deep learning models like PassGAN to the password guessing problem. In this paper, the author starts by stating various kinds of rule based methods

available for cracking passwords and later compares the results of using these methods with respect to that of PassGAN.

Moving on to the architecture of PassGAN, this paper uses the Wassertian GANs (IWGANs) with Adam optimizer. Various hyperparameters like batch size, number of iterations, output sequence length, size of the input vector have been stated. IWGAN is implemented using ResNets since ResNets continuously reduce the training error as the number of layers increases. Residual Blocks are composed of two 1-dimensional convolutional layers, connected with one another with rectified linear units (ReLU) activation functions.

PassGAN was tested on the classic RockYou dataset and linkedin leaked dataset. Figure (2) represents the number of unique passwords generated by PassGAN trained using the RockYou dataset. From the figure, we can infer that the number of passwords matched increased until 199,000 iterations. And after that, increasing the number of iterations reduced the number of passwords matched which implies the model started to overfit. So, after 199,000 iterations, the training was stopped.

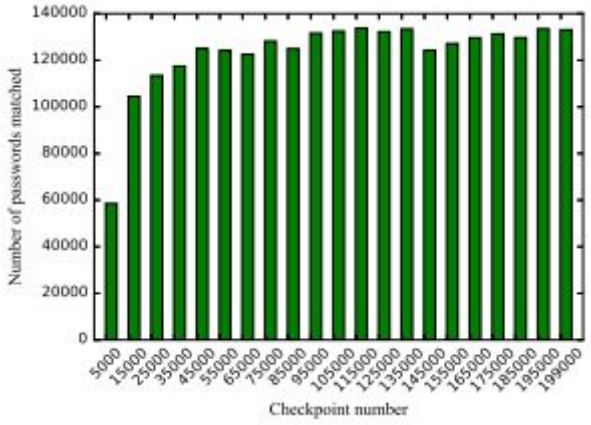

Figure 2 : plot of passwords guesses v/s no. iterations (from paper [3])

When PassGAN was compared with hashcat, the results showed that it was able to match more passwords than hashcat and the most important observation was that PassGAN was able to guess passwords from a dataset different from the one it was trained on.

The paper concludes by stating major advantages of PassGAN over rule based approaches and in which situations rule based methods are preferred over PassGAN. It has been observed that rule based methods were able to outperform when the number of guesses allowed to crack a password was small. However, the main downside of rule-based password guessing is that rules can generate only a finite, relatively small set of passwords. In contrast, PassGAN was able to eventually generate a large number of passwords. As a result, the best password guessing strategy is to use multiple tools since it was observed that by combining the output of PassGAN with the output of the Best64 rules, 50.8% more passwords could be matched.

3.4 Password Guessing using Deep Learning, 2019

The paper introduces various kinds of data breaches and the possible causes for the same. Then, it investigates how well the PassGAN model performs when trained on a large dataset. PassGAN was trained on the Russian email -based dataset and was found that PassGAN did not perform very well when trained on the dataset. After providing suitable explanations for the above mentioned result, the paper concludes by stating some key points to be remembered while training GANs on any type of data.

The paper first introduces various kinds of data breaches using attacks like dictionary attack, brute force, rainbow table attack, etc. Dictionary attacks can be described as guessing passwords by using a specialized wordlist . Brute force attack uses scripts to generate passwords systematically. But, the downside is that for every increasing character, time taken by brute force to crack password, increases exponentially. Rainbow table attacks store pre-computed hash values so that time taken to crack a password significantly reduces.
Next, the paper explores various causes for data breaches. The most common is weak credentials or choosing the same passwords for multiple websites. In phishing, the attacker tries to disguise as a legitimate entity to gain access to sensitive information. Social engineering involves using manipulating tricks to get sensitive information. Ransomware is becoming increasingly popular where data of an organization is infected, and restricted until a ransom amount is paid.

After the model is trained on the Russian email-based dataset, few interesting results were obtained. The most interesting result was the fact that PassGAN was able to detect passwords based on reverse translation technique, where russian words were written in english. These words may seem gibberish to english-speaking users but they are easily recognized by russians.

Analysing the results, the paper concluded that PassGAN did not perform very well when trained on this particular dataset. Table(3) represents the passwords that were not matched by PassGAN. On analysing passwords from Table(3) , it was found that since there was a lot of variation in the structure of passwords in the dataset , PassGAN could not learn a proper distribution that could represent the passwords in the dataset.

NON-MATCHED PASSWORDS

erzhikw	123ko1	trushw	m7802
pirat1501	tj987654	5555223a	lgvi2403q
tybijr123	1842825	Dania200	290292
Aleksrif	tera7	avril509a	m7zenswaq
O7kyr	countesS	menotyou0	nbemej
050613na	babachk	110764m	OOO959881
ixodes23	q12445q	home333	91298123
tapehipu	Nelson09	dbhub	sek790i26
R1der007	13028310q	2403lochs	JersecK
vorovauka	n4055071q	g6l6d6	9KABG2KU

Table 2 : list of non matched passwords (from paper [4])

PROJECT REQUIREMENTS SPECIFICATION

4.1 Introduction

Adversarial password cracking is a method to generate passwords which could be used to determine the strength of password and also used instead of conventional methods. Complete requirements for this project is clearly specified in the following paragraphs.

4.2 Project Scope

The proposed model can be used as a security measure where system administrators can check for easily crackable passwords. It can find its applications in checking/suggesting password strength.

The model can substitute other password cracking tools for generating passwords since it does not require knowledge of the dataset. Also the samples generated are not limited to a password space.

4.3 Product Perspective

Conventionally the password cracking tools use brute force, dictionary based and rule based generation of passwords. Creating rules requires human expertise and is time consuming. Instead GANs could be used which learns from the data and generate passwords which need not know the domain knowledge.

4.4 Product Features

- It generates password guesses which could be used in password cracking
- It can tell whether the passwords is strong or weak

4.5 User Classes and Characteristics

4.5.1 Admin

Responsible for training the model

Has access to generate passwords

Has access to view generated passwords

Check the strength of password

4.5.2 User

Check strength of password

4.6 Operating Environment

- This system requires python, tensorflow and keras to function.
- Presence of fast gpu ensures faster training however is not mandatory.

4.7 General Constraints, Assumptions and Dependencies

The assumptions made :

- The training dataset quality is good and volume is high
- Fast cpu/gpu is available to train faster

Dependencies :

- The model uses tensorflow and keras inbuilt libraries for implementation.

Constraints :

- The performance of the model is constrained by the amount and quality of the dataset being used to train the model.

4.8 Risks

- Since an ML model is used, the model could overfit after training for a large number of iterations.
- Making changes in the model consumes a lot of time since training the model requires a large amount of time.
- ML model has high resource requirements and there is a possibility of resource overflows.

4.9 External Interface Requirements

4.9.1 User Interfaces

- Required screen formats with GUI standards for styles.

4.9.2 Software Requirements

- Python
- Tensorflow
- Keras

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Introduction

System requirement specification contains the description of requirements of software that needs to be fulfilled. The requirements can be functional as well as non-functional. Functional requirements indicate the functionalities that need to be necessarily incorporated into the system and the non functional requirements indicate the various quality constraints that the system must satisfy. The following section contains the description which contains functionalities of the application, interaction of users with various components of the application. This is followed by the functional as well as non-functional requirements of the application.

5.2 Description

5.2.1 List of functionalities supported by the application

- There must be a pre-processing module before the input is fed into the model so that only quality passwords (passwords that represent the distribution of data) must be fed into the model.
- The application enables the training of Generator and Discriminator models by updating weights after every epoch.
- There is a functionality for users to determine the strength of the input password by calculating the number of trials before arriving at the correct password.
- The application also enables admin to tune the hyperparameters of the model in order to achieve more efficiency.

5.2.2 Interactions between user and various other components

When the admin wishes to train the model, he/she should have the possibility to choose the dataset they want to use. There should be an option to split the dataset to train set and test set. They should have the option to choose how many epochs to train on. Once the model is trained they should be able to check how many of the passwords present in the test

set/dataset was the model able to generate. They should have the option customise the model later if required. They should be able to train the model using different datasets. Admin and normal users should be able to use this software to generate as many passwords required. Software should provide an easy to use user interface for this purpose. There should be a provision to export the generated passwords which can be used according to the needs. They should be also able to check the strength of the password. Strength of password should be determined on the likeness of the password being guessed.

5.3 Functional Requirements

- Provision to train if new dataset is provided
- Interface for admin
- Provision to customise the model
- Check the strength of password
- Export list of generated passwords
- Interface for users

5.4 Non Functional Requirements

5.4.1 Usability

The frontend / GUI must be designed in such a way that it is easy to use, and for every action the user performs, there must be a valid response from the back end. Also, the front end must be detailed so that even a user without deep technical knowledge, must be able to appreciate and understand the overall working of the system.

5.4.2 Reliability

Since ML is used, the model must be able to provide accuracy measures to prove that it is reliable. Also, the results of the proposed model must be compared with other existing / new models to interpret the performance of the proposed model. Also, the limitations of the model and the various conditions in which the model will outperform / underperform must be stated clearly.

5.4.3 Performance

For any testing instance, the model must provide results as fast as possible. The model must be trained on a high speed GPU so that the training time can be minimised. While training, the output / state of the model must be saved after some number of epochs so that later, training can be resumed from intermediate epchs as well.

SYSTEM DESIGN

6.1 Introduction

This is a high level design document for ‘adversarial password cracking’. This is a generative adversarial network model which needs to be trained on a list of passwords after which it will be able to generate more likely password guesses. This model can also be used to determine the strength of the password as well.

6.2 Current System

Currently there are various methods of generating password guesses like rule based generation which is time consuming and requires expertise for creating rules. There are also gan models, we aim to generate more likely passwords.

There are various methods to check the strength of password, this also uses some rules, checks for patterns, etc. This model once trained will be able to guess the likeliness of password being cracked. This can be used to determine the strength of the password.

6.3 Design Considerations

6.3.1 Design Goals

The design goals of our project are:

- The model must be able to generate a large number of passwords unlike the rule based methods where the number of passwords generated is limited by the number of rules defined.
- The model must generate more realistic passwords and outperform all of the current rule-based systems.

The principles of design are :

- The software should be designed in such a way that it accommodates and adjusts to the changes done in different phases of software development.
- The design process should not reinvent the wheel. So, we plan to use inbuilt keras modules whenever possible for deploying machine learning models.

The guidelines for project development are :

- The project must be developed using agile methodology so that based on the outcomes, we can make necessary changes to the project at regular intervals.
- Comprehensive validation/testing must be conducted against the specified metrics so that it is absolutely clear if deliverables meet the requirements.

- Extensive documentation and maintenance of the code of the project using version control systems.

6.3.2 Architecture Choices

Some of the alternate architecture choices that can be considered are :

Using autoencoders (or any other generative models) instead of GANs to generate passwords as both are unsupervised learning methods.

Pros of using autoencoders over GANs :

- Autoencoders project the input data to a lower dimension and transform it back to the same shape while GANs do not lower the dimension of the data.

Cons of using autoencoders over GANs :

- Autoencoders are used to restructure the input data and cannot generate new samples while GANs actually learn the distribution of the input data in order to generate new samples of input data.
- Autoencoders introduce a deterministic bias whereas GANs do not introduce any bias.

Using Supervised models instead of GANs to generate passwords.

Pros of using Supervised models over GANs :

- Training a GAN requires finding a Nash Equilibrium. Nash equilibrium is not guaranteed and sometimes model parameters may not converge if nash equilibrium does not exist.

Cons of using Supervised models over GANs :

- The complexity of GANs is less when compared to supervised learning models and also it is very difficult to get labels for leaked passwords dataset.

By considering the above mentioned points, using GANs is the best architectural choice because it can generate passwords by learning the input distribution and also does not require a labelled dataset.

6.4 High Level System Design

Following are the various kinds of components of the project :

- Model component

  This is the core component that is responsible for implementing the GAN model. This is divided into sub components called Generator and Discriminator.

- Generator component

  This component takes random distribution as input and generates passwords after the completion of training. This has methods called generate() and updateWeight() and it interacts with the Discriminator component.

- Discriminator component

  This component takes the output of the Generator component and real samples from input distribution and classifies passwords as real or fake. This has methods called classify() and updateWeight().

- Client / User component

  This component uses functionalities provided by the model component for checking the strength of password input by the user.

- Admin Component

  This component is mainly responsible for providing input data , managing the training process (like customising the hyperparameters) and verifying the generated passwords. Admin component uses functionalities provided by model component for training and evaluation.

The logical data flow of the project is represented by the below diagram :

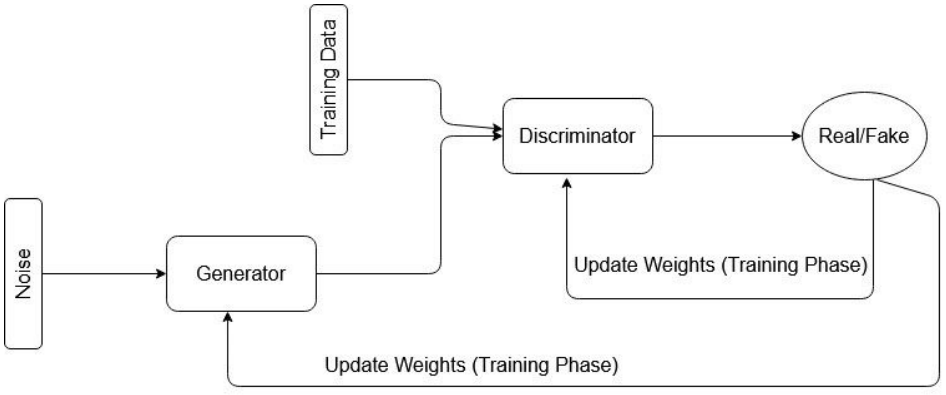

Figure 3 : Logical workflow of the project

Storage and other information

- The model is trained on a kaggle kernel with GPUs since the amount of data is huge and it also reduces the training time. Therefore, the input dataset is also uploaded to the kaggle kernel.

6.5 Design Description

The overall project is broken down into 5 classes :

- User : User module has functionalities to check password strength and generate password.
- DataSet : This class provides the data and also deals with pre-processing the same.
- GAN Model : This is a generalization of Generator and Discriminator classes.
- Generator : This consists of methods for training the generator model.
- Discriminator : This consists of methods for training the discriminator model.

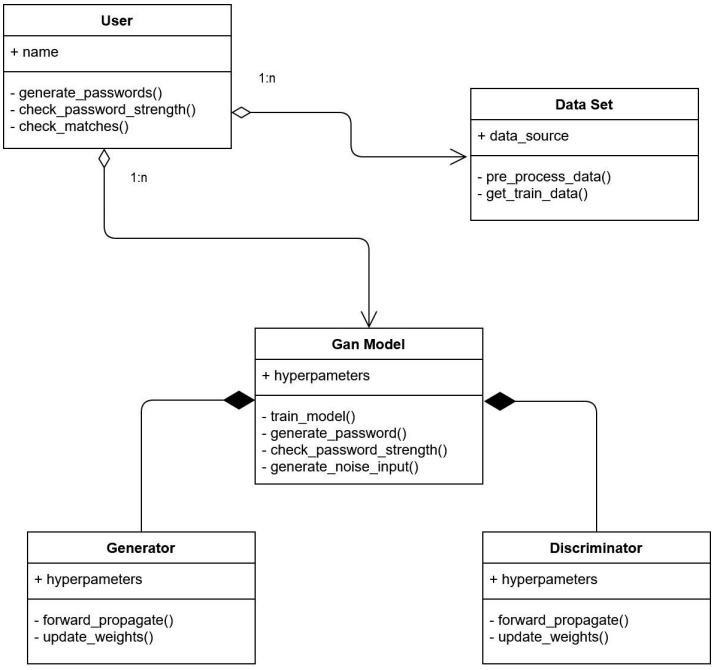

Figure 4 : Master class diagram for the high level design of the project

6.6 User Interface

The following indicates the main functionalities to be implemented as a part of User Interface:

- Provision to process the data for training
- UI for training data
- UI for generation of passwords
- UI for checking the strength of passwords

CONCLUSION OF CAPSTONE PROJECT PHASE - 1

In the first phase of the capstone project, after understanding the importance and use case of password cracking, a detailed literature survey was conducted. From the literature survey, it was evident that machine learning models could be applied for generating password guesses. On researching more about PassGAN, it was found that PassGAN could significantly outperform the current rule based methods and the best results are achieved when PassGAN is combined with other techniques. Also, in one of the papers, checking password strength using PassGAN was suggested as future work. Next, the project requirements document was designed that consisted of project scope, features, constraints / dependencies / assumptions / risks involved in the project development.

Later, the system requirements document was designed that consisted of details about functionalities of various components of the system, their interaction with the user and also the functional and nonfunctional requirements of the application.

After designing the project and system requirements, a high level design document was designed that indicated the overall design which also included a master class diagram for the system to be implemented. Suggestions given at the end of every review was considered and suitable changes were incorporated accordingly. To conclude, in the first phase we got a clear understanding of scope of project, background work done, functionalities to be implemented, and also got an idea about the expected results after the project is implemented.

PLAN OF WORK FOR CAPSTONE PROJECT PHASE - 2

- Low level design
- Implementation of the modules
- Integration of the modules
- Assessing the results
- Further research if result not satisfactory
- Optimizations
- Comparison with other techniques
- Create user and technical manual

REFERENCES/BIBLIOGRAPHY

[1] Suman Nepal, Isaac Kontomah, Ini Oguntola, Daniel Wang Department of Electrical Engineering and Computer Science, MIT Cambridge, '**Adversarial Password Cracking** '

[2] Annie. Chen, '**Presenting New Dangers: A Deep Learning Approach to Password Cracking**', 2018.

[3] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando PérezCruz. 2018. PassGAN: **A Deep Learning Approach for Password Guessing**. In NeurIPS 2018 Workshop on Security in Machine Learning (SECML'18). Montreal, CANADA (colocated with NeurIPS 2018).

[4] Laxmi Ahuja , Vernit Garg , Amity University, '**Password Guessing Using Deep Learning** '