BIPASS: Password cracking Using Bi-Directional Generative Adversarial Network

Sanjay Murmu, Harsh Kasyap and Somanath Tripathy

Department of Computer Science and Engineering,
Indian Institute of Technology, Patna, India
{xx, harsh_1921cs01, som}@iitp.ac.in

Abstract. Password is the most prevalent and reliant mode of authentication, so the attacker's target point is to guess the correct password. Therefore, password generation has become crucial. Passwords used and generated are not secure enough. Deep networks and adversial created samples are mimicking the unthinkable. The overuse of social networks has exposed our user profiles, which increases risk based on individuals and demands personalized models. We have proposed a novel deep learning approach using a bi-directional generative adversarial network "BIPASSGAN" to generate personalized passwords that may be at risk. We have used One-class SVM for determining the strength of the same.

Keywords: Password Machine Learning Transfer Learning Deep Learning GAN PASSGAN BIGAN One-Class SVM.

1 Introduction

Passwords have been the primary source of authentication since we entered the digital age. After a long technological evolution of various other authentication techniques, we still rely on the very same passwords. This over-dependence on passwords requires human efforts and creativity for generating new ones and keeping them secure at the same time. In this machine learning and data-centric era, we often talk that privacy is a myth. With cheap computation and readily available resources, it has become effortless to crack passwords. It introduces requirements beyond human creativity to generate secure passwords. Above all, we have thought about personalizing this scheme. The prevailing models are generalized and may prove strong for one case and weak for the other. Transfer learning can come to rescue, and users can have their personalized models. They can train the model with their specific information on top of an existing model and determine strength of their password.

The above study motivates for designing a new novel approach for generating passwords and very same time checking its performance and strength. So, why the human chosen password is still prevalent. It is easy to implement, and no special hardware is required. Human thinks way different beyond machines. But our thinking is getting exposed over the years through excessive use of the internet. Machine and deep learning techniques have become smart enough to match

our thought. It has brought issues into existence that passwords have become easy to guess. We tend to use the same password as they are hard to remember and difficult to type correctly. It increases the probability of guess and attack.

Before the advancement of machine learning and deep learning, state of the art for guessing attacks were brute force and dictionary attack. In brute-force [**?**] the attacker tries all the password combinations. With this attack, guessing a password is guaranteed but not feasible because of the time taken. The other one is the dictionary attack, as the name suggests, the dictionary may be a collection of word lists which are common sources for users to choose mnemonic passwords.[**?**] Researchers enhanced guessing attacks using probabilistic context-free grammar(PCFG).[**?**] It takes the probability of the frequency of the structure of the password and guesses a password. For example, the guess "password123" may be more probable than the guess "P@ssW0rd!23" depending on the password creation policy and user creativity. In 2016, Melicher et al. proposed that neural network models that can generate human-created passwords.[**?**] Recurrent neural networks can learn from sequences. Researchers use Long short-term memory to avoid gradient vanishing problem. [**?**] All these machine learning algorithms work well.

Still, they all have the same limitations as they can only generate a subset of the password possible and it depends on the knowledge they have extracted from the leaked password. Generative Adversarial Network(GAN) has been widely used in recent years for the generation of new and unseen samples in every corners of research. [**?**] With the help of this generative model (Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, Fernando Perez-Cruz) proposed PASSGAN.[**?**] GAN takes large no of epochs and hence long time for generating passwords. Bi-Directional GAN is an extension of GAN and performs faster in comparison. [**?**]. In GAN, we find the distribution of the data set P(w). With the help of a generator and discriminator, we try to create a distribution with noise(o), which will be similar to P(w). We faced the problem as it takes a longer time to converge the distribution. So, we used Bi-Directional GAN in which by adding one more element, we can converge much faster and get satisfactory results on generating new passwords.

Further, We trained a One-class SVM model for checking the strength based on all the passwords generated based on training over existing and generated passwords. And, We emphasis on personalized model training such that it proves efficient for individuals.

The remainder of this paper is organized as follows. Section 2 discusses the Background and Related Works. Section III summarizes the preliminaries, methodology and explains the proposed work. Section IV lists the experiments and results. Section V concludes and briefs the scope for future work.

2 Related Work

Many authors have proposed and studied password guessing, generation techniques. Earlier, They used brute-force techniques and lexical grammar for this

purpose. With cheap resources available and advancements in deep learning, guessing became easier. Recent models and works analyzed and emulated human thinking and creativity to generate many similar passwords as we can. Nowadays, more exposed profiles have resulted in frequent targeted attacks. It makes the recent works susceptible, ineffective, and motivates us to study them. Many researchers try to learn the pattern of passwords and try to memic passwords generating new or similar kinds of passwords. Researchers used natural processing tools to generate a password. Previous tools include using Markov model [**?**] to improve the dictionary-based attack.

Brute-Force and Dictionary Attack: The oldest password guessing tools are brute force attack [**?**] and dictionary attack [**?**].It can also be categorized as an offline and online attack. The attacker obtains the hash of the password of the target user and tries to get the plain text by guessing iteratively in an offline attack. Thousands, millions of iteration may be required. It is an example of a brute force attack where the attacker tries all possible passwords until it matched the original password. It takes an ample amount of time, and also requires a huge amount of storage. In the dictionary attack, the attacker adds some input with a common password based on some rules and tries to guess the password adding new words systematically by applying pre-selected mangling rules. For a dictionary attack to be successful, it requires the original word to be in the attacker's input dictionary, and for the attacker to use the correct word-mangling rules. The main challenge to make offline password attacks is to generate an ordered list of password guesses p1,p2,p3,p4, and so which matches the original password. An online attack happens when the attacker uses some API to enter password guess against some account.

Markov model: This model uses the idea of a transition between the characters. This idea itself becomes the limitation of this model. It reaches a lower converge rate of testing passwords while generating the same length password. It is hard to choose the effective word mangling rules for a dictionary attack. Omen, an improvement over this model [**?**] adds ranking of passwords for faster cracking. Narayanan and Shmatikov [**?**] used Markov models to generate probable passwords.

Probabilistic context-free grammar model: This approach generates password structures in the highest probability order. The word mangling rules are automatically created. It takes a keyword and multi-word pattern into account. [**?**] Using PCFG in [**?**] Authors observed that not all guesses have the same probability of cracking a password. For example, the guess "password123" may be more probable than the guess "P@ssW0rd!23" depending on the password creation policy and user creativity.

Deep Learning: Researchers also used the concept of neural networks to crack passwords. They created a Recurrent Neural Network[**?**]. It faces an issue of vanishing gradient problem, and it is hard to choose the best n-gram split. It was the first neural network approach for cracking the password. Long short term memory is used as an advanced technique of RNN to solve vanishing gradient problems to solve this problem. [**?**]. A hierarchical semantic model is based

on semantic analysis, which analyzes patterns and semantics of the dataset. It has been tested on the Chinese passwords dataset, which is a collection of leaked passwords from several websites, excluding CSDN passwords. The CSDN passwords are reserved for model testing. We segment passwords into meaningful words and meaningless strings. A neural network is trained with these segmented passwords. [**?**] In 2016, Melicher et al. proposed that neural networks could model human-created passwords. They used neural networks for password strength measures. [**?**] In this model, each password is split in n-gram (best result when n=3), and the n-gram split data is forwarded to the recurrent neural network to generate a new password.

3 PRELIMINARIES

Password data set: The data we have used to generate a new password is the leak RockYou data set. We have taken a 10,00,000 password. we split ed the data in 80 percent for training and the remaining 20 percent for testing.
Data cleaning: Many data in the data set are still in the hash value. Many passwords are repeated and many are contain non-ASCII characters .we have to remove the non-ASCII character from the data set. Their many passwords of the length of 4 characters .we have taken the mini mun password size 6 and the maximum password of 14 characters .most common password around 60 percent password contains lower case letters.
Data conversion: As we split the data set into two-part (a) Training (b) Testing. We take the training data and convert it into an image of the 7*maximum size of the password. To make the dimension of all images the same we set maximum size as 14 by padding space.First we have taken all the number(0-9),upper case character(A,B,..Z),lowercase character(a,b,c,..,z) and special character(@,+,-,..) total 92 including space.Assign every character of the password with a number. Then we have taken each character from a password and take it decimal representative and convert it into a 7-bit binary representation. We have done this work for all the passwords in the training set.

4 Generative Model for password generation

In this section, we will discuss the generative model. Generative Adversarial Network has been recently introduced to train a generative model to generated new data. It consists of two adversarial networks :(a) Generative model G that captures data distribution, and (b) Discriminative model D that estimate the probability that sample have come from training data or generative model. Both G and D could be multi-level perceptron. In Generative adversarial network generator and discriminator play thief and police game where the generator model tries to fool the discriminator and the discriminator try to discriminate. Initially, the discriminator discriminates easily but as the round increases generator network started to update its parameter.

Generator Model: The main motive of this model is to generate the input distribution as real as possible. Eg-if we take an image from the data set this model to try to learn so that it creates a realistic image of real data.
Discriminate Model: The main motive of this model is to identify the fake image produced by the generator from the real data.
GAN training process:

- we take some noise from a random distribution and feed it to the generator.
- we take the generated image and real image to discriminator alternatively.
- the discriminator is a binary classification neural network so it calculates loss for both the fake data and real data and combines them to find the total loss.
- The generator G also calculates the loss from its noise as Gloss.
- both the loss is sent to their respective network to adjust their parameter.
- Apply any optimization algorithm, we have used Adam Optimizer.

GAN's Objective function:

$o \rightarrow NoiseVector \quad G(o) \rightarrow Generator\ output$
$w \rightarrow Training\ sample$
$D(w) \rightarrow Discriminator\ output\ for\ real\ (0,1)$
$D(G(o)) \rightarrow Discriminator\ output\ for\ fake\ (0,1)$

At Discriminator:
$D(w) \rightarrow\ maximized$
$D(G(o)) \rightarrow\ minimized$

At Generator:
$D(G(o)) \rightarrow\ maximized$

Loss Function:
At Discriminator D:

$Dloss_{real} = log(D(w))$
$Dloss_{fake} = log(1 - D(G(o)))$
$Dloss = Dloss_{real} + Dloss_{fake} \ = \ log(D(w)) + log(1 - D(G(o)))$
the total cost is
$\frac{1}{m} \sum_{i=1}^{m} log(D(w^i)) + log(1 - D(G(o^i)))$

At Generator G:

$Gloss = log(1 - (D(o)))\ or\ -log(D(G(o)))$
the total cost is
$\frac{1}{m} \sum_{i=1}^{m} log(1 - (D(o^i)))$ or

$\frac{1}{m} \sum_{i=1}^{m} -log(D(G(o^i)))$

Total Loss of GAN:

$$V(D,G) = E_{w \sim pdata(w)}[logD(w)] + E_{o \sim po(o)}[log(1 - D(G(o)))].$$

Bi-directional GAN just extends the GAN by adding the third component. The third component is the encoder. the work of the encoder is to map the data from data space w to latent space o.the objective of the generator remains the same whereas the objective of the discriminator is altered to classify between the real sample and the fake sample and additionally between a real encoding and a fake encoding.

Generator(Decoder):
Assume that we have a prior belief on where the latent space o lies:

$$P_O(o)$$

Given a draw from this latent space the generator G, a deep learner, outputs a synthetic sample.

$$G(o|\theta_G) : o \rightarrow w_{synthetic}$$

Encoder: This is the inverse of the generator. From a given data space the encoder E outputs a real encoding.

$$E(w|\theta_E) : w \rightarrow o$$

Discriminator: The discriminator classify if a sample is from the real data distribution.

$$P_W(w)$$

or the synthetic data distribution

$$P_G(w|o)$$

classify whether an encoding is real

$$P_E(o|w)$$

or synthetic

$$P_O(o)$$

BIGAN Objective Function:
$\min_{G(o;\theta_G),E(w;\theta_E)} \max_{D[(w,o);\theta_D]} V(D[(w,o) : \theta_D], G(o;\theta_G), E(w;\theta_E) = \mathrm{E}_{w\sim P_W(w)} E_{o\sim P_E(o/w)} log[D[(w,o);\theta_D]]$
$\mathrm{E}_{o\sim P_O(o)} E_{w\sim P_G(w/o)} log[1 - D[(\hat{w},o);\theta_D]]$

Training Bigan:
We trained our Bigan model with the train password data which we have converted into an image with a dimension of 7*maximum size of the password i.e 14 in our case. We converted each password into an image and feed it to the discriminator with the decoder of it as well as the generated image with its vector representation alternatively. It took 6 hrs to train the model on the RockYou data set with batch size 64 on the Ubuntu cluster.

Table 1. Representing text into binary

Password	Binary Representation
sanjay	['0110110', '0100100', '0110001', '0101101', '0100100', '0111100']
sandip	['0110110', '0100100', '0110001', '0100111', '0101100', '0110011']
rockyou	['0110101', '0110010', '0100110', '0101110', '0111100', '0110010', '0111000']
devil	['0100111', '0101000', '0111001', '0101100', '0101111']
passwor	['0110011', '0100100', '0110110', '0110110', '0111010', '0110010', '0110101']
Riya	['0011011', '0101100', '0111100', '0100100']
123456	['0000001', '0000010', '0000011', '0000100', '0000101', '0000110']
8820337820	['0001000', '0001000', '0000010', '0000000', '0000011', '0000011', '0000111', '0001000', '0000010', '0000000']

5 Creating Password using BipassGAN

After the password image is given to the Bigan, the model trains itself and generates an image with a dimension of 7*14 as output. During the generation the value range from 0 to 1.To make it as binary representation we give a threshold in which less than 0.5 will be represented as 0 and greater as 1. Then we take 7 bit sequentially and convert the binary representation to decimal. Then with the help of our character to decimal representation, we convert the decimal to its character representation. As all the passwords will be of 14 sizes because we padded with space.so after the generation of the password we remove the padded space from the password. In our experiment, we have taken 64 batch sizes with Adam optimizer with a learning rate of 0.002.

Evaluating the password:

After the generation of passwords, with the help of set theory, we can find a unique password generated by the model. First, we take all the test data into a set to find the unique test password. Then we compare the generated password with the test data.we have significant results compared to the first deep learning approach.

Table 2. Result of passgan using the Rockyou data set in 199000 iteration

Password Generated	Unique Password	Password Matched(1978367 unique sample)
10^4	9738	103(0.005)
10^5	94400	957(0.048)
10^6	855972	7543(0.381)
10^7	7064483	40320(2.038)

6 Password Strength Evaluation:

With the help of one-class SVM, we have evaluated the strength of the password. We have trained the one-class SVM model[**?**] with the generated password that we have generated from the Bigan. After training the model with the password we test it a human-created password and check whether the password is weak,

Table 3. Result of bipassgan using the My space data in 160000 iteration

Password Generated	Unique Password	Password Matched(145588 unique sample)	Percentage
10^4	9878	247	0.16
10^5	97323	1123	0.76
10^6	971712	6234	4.28
10^7	8732545	32571	22.3

Table 4. Result of bipassgan using the rockyou data set in 160000 iteration

Password Generated	Unique Password	Password Matched(152848 unique sample)	Percentage
10^4	9978	287	0.18
10^5	99473	1334	0.87
10^6	978976	7318	4.78
10^7	8324568	39648	25.8

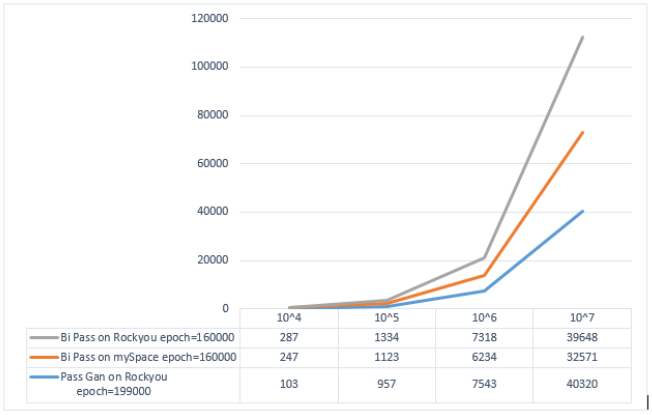

Fig. 1. Compare Between PASSGAN AND BIGAN

normal or strong.

Support Vector Machine: Let's take look at the traditional two-class support vector machine(SVM).Consider a data set = $(w1, u1), (w2, u2), \ldots, (w_n, u_n)$ $where$w$_i$ is the input data point and u_i indicate the class. The property of SVM is that it helps in separating data into a class. Datapoint which cannot be separated by a straight line in original space is lifted to higher space where there can be a "straight" hyperplane that separates the data points of one class from another. When that hyperplane would be projected back to the input space, it would have the form of a non-linear curve. It is one of the best algorithms for classification problems. But as we have no label data we unable to use a two-class support vector machine, for this reason, we take the help of a one-class support vector machine.

One class support vector machine: In one-class SVM, the model is trained on data that has only one class that is normal data. It infers the properties of normal cases and with the help of these properties can predict which are unlike

Table 5. Few example of Feature Extraction for one class SVM

Password	char length	numeric	alphabet	specials	vowel	consonants
password	8	0	8	0	2	6
boncev	6	0	6	0	2	4
dWrraq	6	0	6	0	1	5
ojscatgxp	9	0	9	0	2	7
beoYobs	7	0	7	0	3	4
G48gi	5	0	3	0	1	2
1aWdWpk”	8	1	6	1	1	5
rpgscqsF	8	0	8	0	0	8
ohi!ewg8J	9	1	7	1	3	4
NOCD88	6	2	4	0	1	3
pqblmiq/	8	0	7	1	1	6
wanoe47	7	2	5	0	3	2
jikiqa	6	0	6	0	3	3
WioY00	6	2	4	0	2	2
qiskn	5	0	5	0	1	4
9uH”8w90	8	4	3	1	1	2
qomeccdltX	10	0	10	0	3	7
Wnqjrcl	7	0	7	0	0	7
iran85	6	2	4	0	1	3
ambirdf5	8	1	7	0	2	5
clile!5	7	1	5	1	1	4

the normal example. with the help of this, we have measured the strength of the password. One class SVM consists of outlier and inlier.when we enter a new password for testing for prediction, it returns the score from which we calculate the distance from the outlier and inlier and determine whether the strength of the password.

Result: First, we have train our SVM model with 70 percent of RockYou leaked password and tested the model with a summation of 30 percent of RockYou and 30 percent of my space. We found that a number of a strong password are 11983(16% percent), a number of the medium password are 23257(31% percent), a number of the weak password is 39618(52% percent).In 7 and First Experiment . we have shown some results of our first test and the bar chart.

In the second we have trained our oen class SVM model with 70 percent of RockYou leaked password and 70 percent of my space leaked password and tested the model with a summation of 30 percent of RockYou and 30 percent of my space. We found that a number of a strong password is 18912(34.47% percent), the number of the medium password is 16130(29% percent), the number of the weak password is 19186(34.97% percent).In 8 and Second Experiment. we have shown

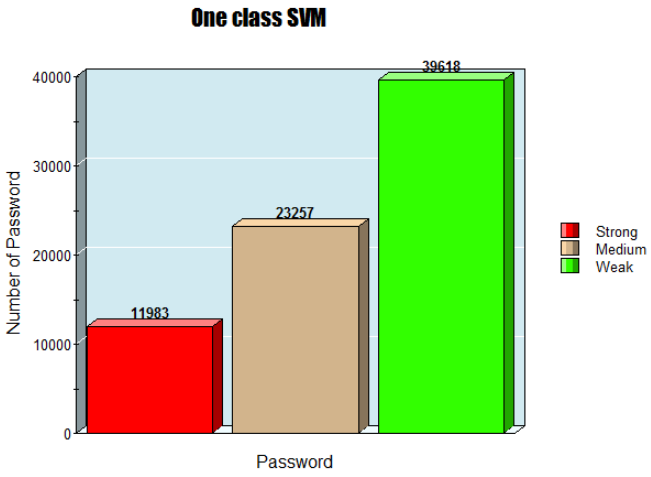

Fig. 2. First Experiment

some results of our second test and the bar chart.

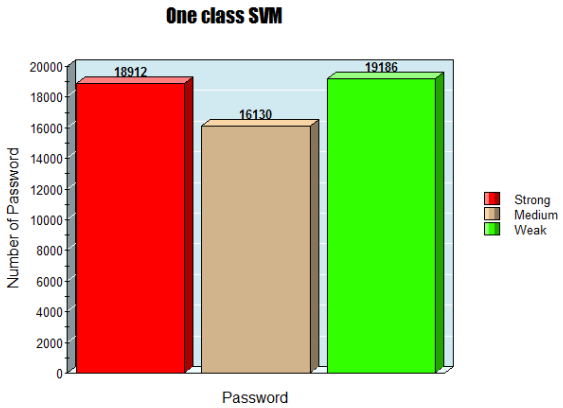

Fig. 3. Second Experiment

In the third, we have trained our one class SVM model with 60 percent of RockYou leaked password and 60 percent of my space leaked password and 60 percent of the generated password by BiGAN and tested the model with a summation of 40 percent of RockYou and 40 percent of my space. We found that the number of a strong password is 7429(21% percent), the number of the medium password

is 4284(12% percent), number of weak passwords 23524(66% percent).In 9 and Third Experiment. we have shown some results of our third test.

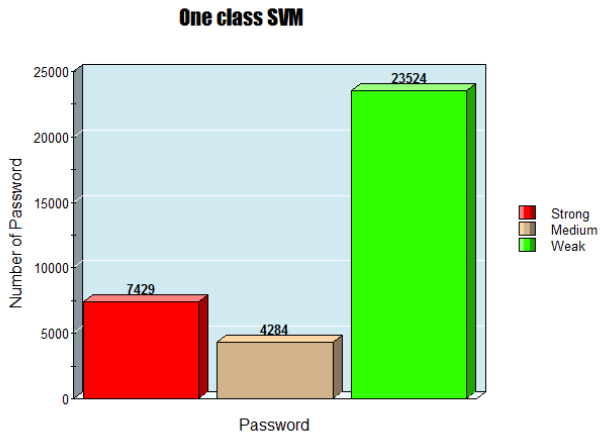

Fig. 4. Third Experiment

7 Password Strength Metre:

We have created a password strength meter with the help of a one-class support vector machine. with the help of the tool called PARS by Georgia Tech university, we have created a data set of passwords with their strength. We then extracted features from the password. The features are the length of the password, number of vowels, number of special characters, number of the alphabet, number of consonants and number of numerical. We have done two testings with the data set 1.in the first experiment we have trained the one-class SVM with 70 percent of data set and test with 30 percent of the data set we got an accuracy of 87 percent but when we added the generated data with data set the accuracy drop to 55 percent it shows that we have successfully created a large subset of unique password 2. in the second experiment we give the data set to one class for training. And for testing, we are taking input from the password from the user and give the strength of the user input password. By comparing it with the available it showing the great result.

Table 6. Few example of One class Support Vector Machine Password Strength Checker

Password	The password Strength Checker	OCSVM Password Strength Checker
ANju8820	Very Strong	Weak
Kundan@1994	Very Strong	Good
JAY!@882037820	Very Strong	Good
Bhavendra206	Strong	Good
8963343643	Weak	Weak
25081998@devil	Strong	Good
lucy@!9863	Strong	Weak
Sirdhar123456!@	Strong	Good
lovemysoul	Weak	Weak
liveThelife@1231224	Strong	Strong
netdome8752dxja	Strong	Good
amazon2345da	Good	Good
SANjgaydyatig	Good	Strong
Riya@199578fdjy	Strong	Good
juhiGyani@kichkich	Strong	Strong
Shikhatek+457674	Strong	Good
JyotiKhateN1235	Strong	Good
PiyaDon45678	Strong	Medium
RiyaJay14758788534	Strong	Strong
Abcjgdjkash	Weak	Good
123456789	Weak	Weak
987654321	Weak	Weak

8 Conclusion

We conclude that though we have pass gan the first deep learning approach for generating a password from the leaked data set but with this new approach "BI-PASSGAN" we have generated a password with less epoch and our generated password is closer to human-created password. With the ample amount of passwords, we have made a password strength meter that works on the approach of a one-class support vector machine.

9 Future Works

We can use the concept of transfer learning for tuning Bidirectional GAN. As we trained the bi gan model with a large dataset and it takes a huge amount of time to complete. So we can use the concept of transfer learning to retrain the gan model whenever the new dataset is given. We have used the one-class SVM model for analyzing the strength of the password. Tuning of one class SVM can be found as it has only one class and it hard to tune the best parameter for the one class SVM

Table 7. Password Strength evaluation using one class SVMExperiment no :1

Leaked Password	Strength value	Strength
tmk417	0.901166182	strong
lakachan+	0.37563582	medium
ahlburg	0.0293117	weak
game310	0.21696324	weak
dennyboi$_4$4	0.45553371	medium
detroit3	0.06841822	weak
ickynicky7	0.42851553	medium
muther10	0.42422382	medium
magic7	0.06035618	weak
goodman3	0.36192017	medium
sexyass	0.40710514	medium
jama1ca	0.24684036	weak
clongee3	0.31017658	weak
DCFC07	0.901166182	strong
pitzy101	0.20409851	weak
1freefromyou	0.43597431	medium
pn3rrz	0.901166182	strong
00336alt	0.10046797	weak
schwingy6	0.37160577	medium
mystery07	0.38166591	medium
i,3tez	0.901166182	strong
oscar5	0.1941001	weak
yellow101	0.51695298	medium
babyj1	0.58394568	medium
Malamute1	0.2570875	weak
555then666	0.30653436	weak
9865626	0.06399395	weak
east a dick	0.29386985	weak
cowiejaw!	0.03046954	weak
k6108720	0.08083838	weak

Table 8. Password Strength evaluation using one class SVMExperiment no:2

Leaked Password	Strength value	Strength
tazdevil07	0.32036569	medium
ilovejeff2	0.944243372	strong
ilyricky7.	0.87256775	strong
loveya1	0.933616675	strong
lucille06	0.6604346	medium
corsica1	0.42160191	medium
amljml95	0.900984277	strong
theknot27	0.35216362	medium
*broken	0.34658619	medium
miguel23	0.17761339	weak
pglpgl1	0.900984277	strong
mustang1	0.900624452	strong
ytel123	0.42482441	medium
wiremesh1	0.45775624	medium
tabnjohn1	0.28798839	weak
caly123	0.57988558	medium
radi20nov	0.27680319	weak
bigpimpn	0.34024896	weak
parker07	0.53025506	medium
jpharper	0.65427012	strong
13pizza.	0.11038831	weak
butkus51	0.09481831	weak
lhs2011	0.3550954	medium
angel123	0.931659671	strong
1482east	0.28366696	weak
apples03	0.4823104	medium
basketbal1	0.42195953	medium
savvy5	0.900984277	strong
mon614	0.54370133	medium
.colchones	0.86624533	strong
basket3	0.17903364	weak
chicken2	0.9148074	strong
Snowdrop7	0.20821502	weak
geturight	0.46668662	medium
goaway2	0.16810813	weak
usemos69	0.26655816	weak
ilovematt2	0.940467097	strong
Eagles1+6	0.45644598	medium
holiday88	0.45903139	medium
60nv89f	0.900984277	strong

Table 9. Password Strength evaluation using one class SVMExperiment no :3

Leaked Password	Strength value	Strength
310ja	0.09614718	weak
iceiceice1	0.12260366	weak
demandalow	0.82285819	strong
hawaii0007	0.25389768	weak
z200146841	0.50760163	medium
ILUVU2	0.36769857	medium
fishyp@sbcglobal.net	0.29067678	weak
KOUGIE8	0.07039939	weak
ciaobella	0.62995626	medium
21886	0.12852141	weak
likeomgg	0.10556222	weak
ir96821	0.01239297	weak
ycnigga	0.14521796	weak
123456p	0.90260889	strong
dt3442853	0.901166182	strong
colts18	0.09495989	weak
tabbycat666	0.2435232	weak
8000rpm	0.11821646	weak
leonpisan1	0.28687917	weak
h3110h3110	0.20296103	weak
mikymac	0.11280279	weak
german1	0.49867122	medium
spring1	0.47126069	medium
merna69	0.29389055	weak
singing1	0.77622032	medium
NASTY31	0.31092699	weak
ad1218	0.25958627	weak
hardc0re	0.50476271	medium
beetljui4	0.2192165	weak
downhill1	0.53713512	medium
h13a51	0.900984277	strong
ubby	0.18392788	weak