Information systems. Computer sciences. Issues of information security

Информационные системы. Информатика. Проблемы информационной безопасности

UDC 004.89
https://doi.org/10.32362/2500-316X-2023-11-4-7-15

RESEARCH ARTICLE

Password strength verification based on machine learning algorithms and LSTM recurrent neural networks

Vladimir V. Belikov [1, @],
Ivan A. Prokuronov [2]

[1] *MIREA – Russian Technological University, Moscow, 119454 Russia*
[2] *SFB Laboratory, Moscow, 127083 Russia*
[@] *Corresponding author, e-mail: belikov_v@mirea.ru*

Abstract
Objectives. One of the most commonly used authentication methods in computer systems, password authentication is susceptible to various attacks including brute-force and dictionary attacks. This susceptibility requires not only the strict protection of user credentials, but also the definition of criteria for increasing a password's strength to minimize the possibility of its exploitation by an attacker. Thus, an important task is the development of a verifier for checking passwords for strength and prohibiting the user from setting passwords that are susceptible to cracking. The use of machine learning methods to construct a verifier involves algorithms for formulating requirements for password complexity based on lists of known passwords available for each strength category.
Methods. The proposed supervised machine learning algorithms comprise support vector machines, random forest, boosting, and long short-term memory (LSTM) recurrent neural network types. Embedding and term frequency–inverse document frequency (TF-IDF) methods are used for data preprocessing, while cross-validation is used for selecting hyperparameters.
Results. Password strength recommendations and requirements from international and Russian standards are described. The existing methods of password strength verification in various operating systems are analyzed. The experimental results based on existing datasets comprising passwords having an associated level of strength are presented.
Conclusions. A LSTM recurrent neural network is highlighted as one of the most promising areas for building a password strength verifier.

Keywords: cybersecurity, password strength, supervised machine learning, recurrent neural network, LSTM

• **Submitted:** 11.12.2022 • **Revised:** 01.02.2023 • **Accepted:** 02.05.2023

For citation: Belikov V.V., Prokuronov I.A. Password strength verification based on machine learning algorithms and LSTM recurrent neural networks. *Russ. Technol. J.* 2023;11(4):7–15. https://doi.org/10.32362/2500-316X-2023-11-4-7-15

Financial disclosure: The authors have no a financial or property interest in any material or method mentioned.

The authors declare no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Построение верификатора стойкости пароля с использованием классических методов машинного обучения и рекуррентной LSTM нейронной сети

В.В. Беликов [1, @],
И.А. Прокуронов [2]

[1] *МИРЭА – Российский технологический университет, Москва, 119454 Россия*
[2] *СФБ Лаборатория, Москва, 127083 Россия*
[@] *Автор для переписки, e-mail: belikov_v@mirea.ru*

Резюме
Цель. Аутентификация с использованием паролей является одним из наиболее распространенных способов проверки подлинности в компьютерных системах. Существующие атаки на пароли, включающие в себя, в т.ч. атаки перебора и атаки по словарю, требуют не только защиты учетных данных пользователя на этапе эксплуатации паролей, но и определения требований к паролю, позволяющих повысить стойкость пароля к атакам, минимизируя возможность их реализации злоумышленником. Важной задачей при этом становится разработка верификатора, осуществляющего проверку пароля на стойкость и позволяющего исключить задание пользователем паролей, подверженных взлому. Построение верификатора с использованием методов машинного обучения позволяет алгоритмам самим формулировать требования к сложности пароля в произвольно комплексной форме, отталкиваясь только от инцидентов, имеющихся для каждой категории стойкости списков известных паролей.
Методы. Предложены алгоритмы машинного обучения с учителем: метод опорных векторов, случайный лес, бустинг, рекуррентная LSTM (long short-term memory) нейронная сеть. В эксперименте для предобработки данных применены метод простой индексации символов с последующей обработкой embedding-слоем и метод TF-IDF (term frequency-inverse document frequency). Для выбора гиперпараметров алгоритмов была использована кросс-валидация.
Результаты. Проведен анализ рекомендаций и требований к паролям в международных и отечественных стандартах и возможности их реализации в виде верификатора стойкости пароля в различных операционных системах. Приведены результаты эксперимента на существующем наборе помеченных по уровню стойкости паролей. Проведена их оценка с использованием macro f1-меры.
Выводы. Использование рекуррентной LSTM нейронной сети выделено как одно из наиболее перспективных направлений для построения верификатора стойкости пароля.

Ключевые слова: компьютерная безопасность, стойкость пароля, машинное обучение с учителем, рекуррентная нейронная сеть, LSTM

• **Поступила:** 11.12.2022 • **Доработана:** 01.02.2023 • **Принята к опубликованию:** 02.05.2023

Для цитирования: Беликов В.В., Прокуронов И.А. Построение верификатора стойкости пароля с использованием классических методов машинного обучения и рекуррентной LSTM нейронной сети. *Russ. Technol. J.* 2023;11(4):7–15. https://doi.org/10.32362/2500-316X-2023-11-4-7-15

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

Password authentication represents one of the most common authentication methods used in computer systems [1]. Existing password threats, including brute-force-, dictionary-, and rainbow table attacks, require not only the protection of user credentials during password exploitation, but also the definition of password requirements when setting a password. Such requirements should increase password strength to withstand specified attacks, minimizing the possibility of their being compromised by an attacker [2]. In this connection, an important task involves the development of a verifier checking the password for strength to prohibit users from setting passwords that are vulnerable to cracking [3].

ANALYSIS OF EXISTING APPROACHES TO BUILDING A PASSWORD STRENGTH VERIFIER

The most commonly used Russian standards for setting a strong password are set out [4] in the following documents:

1) National Institute of Standards and Technology (NIST) Special Publication "Digital Identity Guidelines. Authentication and Lifecycle Management"[1];
2) Methodological document of the Federal Service for Technical and Export Control (FSTEC) "Information Protection Measures in State Information Systems"[2].

These guidelines contain recommendations for users when creating passwords, as well as requirements and recommendations for verifiers (websites, software, etc.) that contain a system for checking and processing passwords.

The first document refers to passwords as "memorized secrets." Among the general provisions are the following:

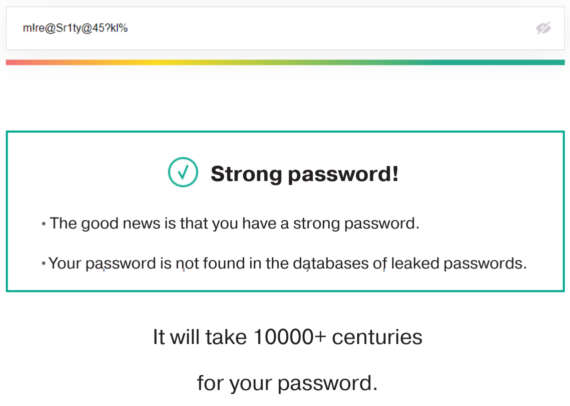
1) memorized secrets (passwords) must be at least 8 characters long if selected by the user;
2) memorized secrets (passwords) randomly generated by the computer or verifier must be at least 6 characters long and may consist entirely of digits;
3) if the computer or verifier prohibits the selected memorized secret (password) on the grounds that it is contained in the previously accepted blacklist of compromised values, the user should select another memorized secret (password).

In the FSTEC document, the recommended minimum password length of 6 characters achieves the requirements of the simplest level (level 4) of personal data security [5].

When processing requests to establish and change memorized secrets, verifiers should compare assumed secrets with a list of frequently used, expected or compromised passwords. For example, the list may include:

1) passwords obtained from the database of cracked passwords;
2) vocabulary words;
3) repetitive or sequential characters (for example, "qqqqqq," "qwerty12345");
4) context-dependent words such as service name, user name, and their derivatives (e.g., "mireastudent," "ivanivanov").

A convenient way to check a password for strength is to use special services such as those offered on the Kaspersky website[3]. This service provides information on password strength, occurrence of the password in leaked databases, as well as how long it would take to crack a particular password using a brute-force attack (Fig. 1).


m!re@Sr1ty@45?kl%

✓ Strong password!

• The good news is that you have a strong password.

• Your password is not found in the databases of leaked passwords.

It will take 10000+ centuries

for your password.

Fig. 1. Demonstration of password verification on the Kaspersky website

The complexity of user-selected passwords can be characterized using the concept of information theory, whose founder is widely considered to be the American mathematician, Claude Shannon. Although entropy, representing a measure of the information capacity of the system, can be easily computed for data with deterministic distribution functions, evaluation of entropy for user-chosen passwords represents a complicated task, which defeats attempts to derive an exact result on this basis. For this reason, for example, the NIST publication uses an approach based primarily on password length.

[1] https://pages.nist.gov/800-63-3/sp800-63b.html. Accessed February 01, 2023.

[2] https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-11-fevralya-2014-g (in Russ.). Accessed February 01, 2023.

[3] https://password.kaspersky.com/ru/ (in Russ.). Accessed February 01, 2023.

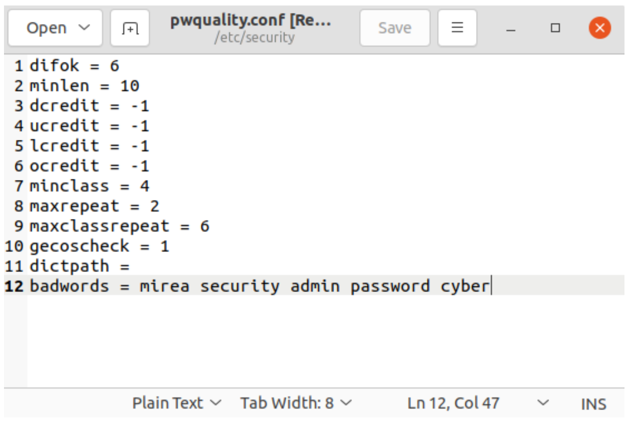
In order to provide password strength testing in different operating systems, special modules are used to evaluate passwords against some criteria that adhere to generally accepted recommendations, as well as offering the possibility of setting such criteria by the system administrator. In general, this is referred to in terms of password policy, comprising a set of rules improving the security of user accounts by encouraging them to use stronger passwords.

Linux operating systems have long used the pluggable authentication module (PAM) *pam_cracklib* designed for verifying passwords by dictionary words [6]. In recent versions of Linux, this module has been replaced by the *pam_pwquality* module based on the *pam_cracklib* module, which is fully backward-compatible with its predecessors. This module offers a simpler password policy approach for verifying that users have conformed with the administrator's password requirements.

The following list of requirements represents an example of a password policy set using the specified module:

1) minimum password length is 10 characters;
2) the new password must contain 6 new characters not present in the old password;
3) the password must contain lowercase letters, uppercase letters, special characters, and numbers;
4) the password must not contain more than two characters in a row;
5) the password must have no more than six consecutive characters from the same class;
6) the presence of GESOC check;
7) prohibiting the words "mirea," "security," "admin," "password," and "cyber."

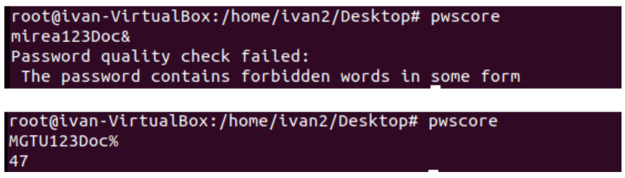
The configuration file */etc/security/pwquality.conf* reflecting the described security policy is shown in Fig. 2.



```
difok = 6
minlen = 10
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
minclass = 4
maxrepeat = 2
maxclassrepeat = 6
gecoscheck = 1
dictpath =
badwords = mirea security admin password cyber
```

Fig. 2. Configuration file *pwquality.conf*

The *pwscore* utility included in the *libpwquality* package can be used for checking the correctness of the specified password policy. An example of testing the password that contains a forbidden word and one that meets all requirements using the *pwscore* utility, which gives a score from 0 to 100, is shown in Fig. 3.



```
root@ivan-VirtualBox:/home/ivan2/Desktop# pwscore
mirea123Doc&
Password quality check failed:
 The password contains forbidden words in some form
```

```
root@ivan-VirtualBox:/home/ivan2/Desktop# pwscore
MGTU123Doc%
47
```

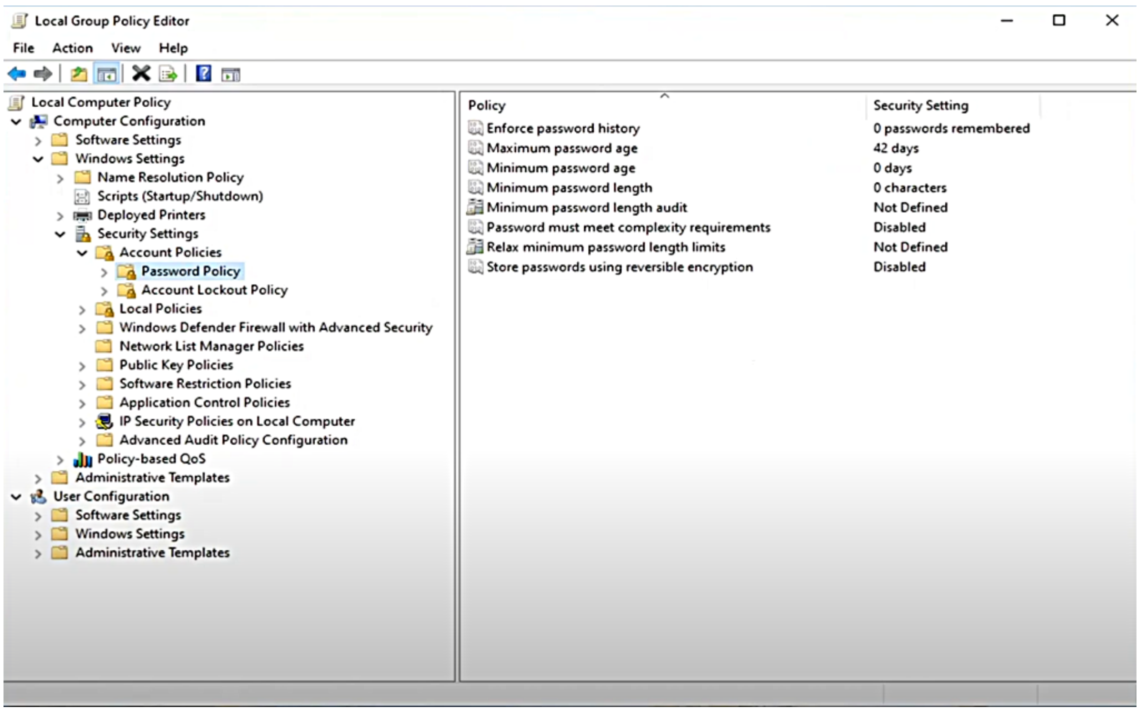
Fig. 3. Password testing

In Windows operating systems (OS), the password policy setting is set in the "Local Security Policy" section of the *gpedit.msc* utility [7] (Fig. 4).

Offering only eight configurable options, Windows OS allows the password policy to be set and changed according to strictness criteria, including and/or combining the set values of these options. One of the most important of these is the "Password must meet complexity requirements" option. Enabling this option specifies that all passwords should meet the following requirements:

1) the password must not contain a user account name or parts of a full user name longer than two consecutive characters;
2) the minimum password length is 6 characters;
3) the password must contain characters from at least three classes: Latin uppercase letters, Latin lowercase letters, numbers, and special characters. In Windows 11, the fourth class requiring any Unicode character is added.

Thus, the analysis of existing approaches to password strength verifier construction shows that the most common solution consists in the allocation of rules focusing primarily on password length and its presence in existing password dictionaries. This solution has a number of disadvantages including the difficulty of implementing more complex requirements reflecting rigorous password-strength properties. In contrast, machine-learning approaches to building a verifier uses algorithms to formulate these requirements themselves in an arbitrarily complex form relying only on incidents that comprise lists of known passwords available for each strength category. One of the properties of this approach is its versatility, since creating and customizing a password policy (number of special characters, number of digits, etc.) for each specific case is not required. The machine simulates a mixture of many password verification algorithms. The trained machine can be used as a separate module (for example, PAM module in Linux) for ensuring that password strength is accurately determined. For example, if a developer is creating some new social network, it is sufficient to teach the machine using a dataset from other popular social



Policy	Security Setting
Enforce password history	0 passwords remembered
Maximum password age	42 days
Minimum password age	0 days
Minimum password length	0 characters
Minimum password length audit	Not Defined
Password must meet complexity requirements	Disabled
Relax minimum password length limits	Not Defined
Store passwords using reversible encryption	Disabled

Fig. 4. Password policy settings window in Windows OS

networks, where the password policy is considered to be effective: TikTok[4], Twitter[5] (banned in Russia), Facebook[6] (banned in Russia), etc. The machine is then built in as a verifier evaluating the password and providing the corresponding information to the user when registering. In this case, if the entered password is given the highest stability score, the user can be sure that his/her password is really stable without being bound to specific requirements and recommendations.

METHODOLOGY AND EXPERIMENTAL RESULTS

The task of password strength verification can be presented as a classification problem where the data object is a password while the class is its strength level.

The dataset of one of the largest password leaks, comprising the hosting site 000webhost [8], is used as the dataset for the experiment. Using the *PARS* tool [9] containing many counters evaluating password strength, the password database and its evaluation are based on three different algorithms implemented by Twitter, Microsoft[7], and Battle.net[8], which were created by an independent developer. The dataset itself contains only the passwords rated equally by all three measures. There are three classes of password scores in the dataset (0 is low, 1 is medium, and 2 is high). The dataset contains unbalanced data: 496 649 passwords rated “1,” 89 662 passwords rated “0,” and 83 113 strong passwords.

The examples of passwords for each strength level are shown in Table 1. The distributions of password length for the whole dataset, as well as for individual strength levels, are shown in Figs. 5 and 6.

Table 1. Examples of passwords from training dataset

Password strength	Password examples
0	wewes19 asdas95
1	oyeleye1 80188063JA
2	JFRTgxTQyNQTh9ZD d7a6AoTMxMw0dLVy

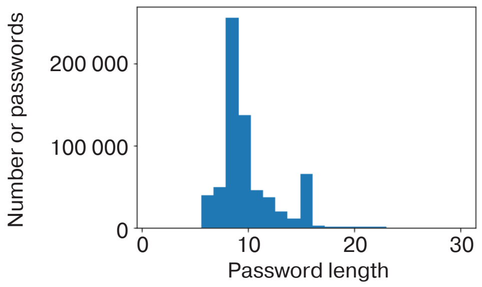


Fig. 5. Histogram of the password length in the dataset

[4] https://www.tiktok.com/. Accessed February 01, 2023.
[5] https://twitter.com/. Accessed February 01, 2023.
[6] http://facebook.com/. Accessed February 01, 2023.
[7] https://www.microsoft.com/. Accessed February 01, 2023.
[8] https://www.battle.net. Accessed February 01, 2023.

Table 2. Learning results evaluated on the training dataset by cross–validation

Algorithm	Found hyperparameter	Macro f1-measure
Support vector machine	Regularization coefficient: 1	0.806
Random forest	Maximum tree depth: 32	0.903
Boosting	Maximum tree depth: 32	0.946

Table 3. Learning results evaluated on the test set

Value	Boosting	LSTM recurrent network
Precision	0.972	0.9995
Accuracy	0.971	0.9994
Completeness	0.971	0.9990
Macro f1-measure	0.971	0.9992

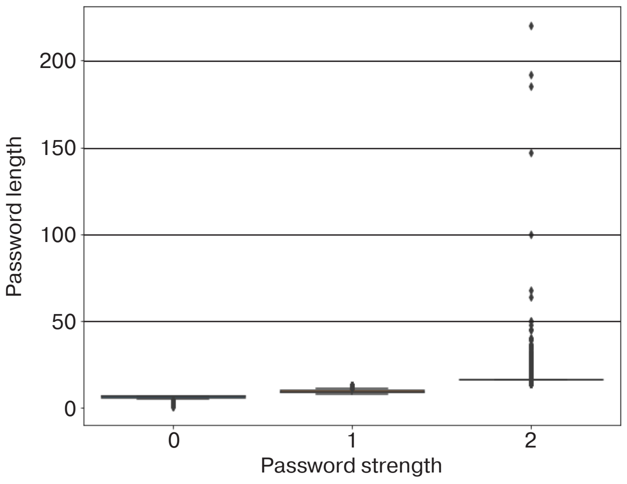


Fig. 6. Distribution of password length depending on strength in the dataset

For the experiment, the set is split into a training set (80%) and a test set (20%) after random mixing.

The term frequency–inverse document frequency (TF-IDF) method [10] is used for representing the text password as a vector of numeric values in order to consider not only the presence, but also the weight of each individual character, as follows:

$$TF = \frac{\text{number of character occurencies in password}}{\text{total number of characters in password}},$$

$$IDF = \log_2 \frac{\text{total number of passwords}}{\text{number of passwords with character}},$$

$$TFIDF = TF \cdot IDF,$$

After applying the TF-IDF method to the set of passwords, a set is obtained where each password has the same length as the size of the dictionary, while each element vector has TF-IDF weight of the character whose serial number in the dictionary corresponds to the serial number of the vector element.

The equalization of the number of instances in each class with an undersampling technique and *k*-fold cross–validation [11] with $k = 5$ is used for learning classical algorithms and for selecting hyperparameters, respectively. The results of learning the algorithms on the training dataset evaluated using the macro f1-measure [12] are presented in Table 2.

At the same time, the use of classical machine learning methods is complicated by the need to transform passwords using TF-IDF algorithm; due to the necessity of recalculating weights when the password set is updated, this is a costly procedure. In addition, these algorithms represent passwords as an unordered set of characters (a simplification used in the "bag of words"[9] approach), which does not consider the mutual arrangement of characters. This can be partially fixed by using bigram- and trigram models, which in turn makes the learning process much more difficult. In contrast, recurrent neural networks work directly with character sequences of arbitrary length [13]. Thus, for a recurrent neural network, the "PASSWORD" and "AWSSODPR" passwords are determined by two different vectors, whereas when using TF-IDF they are determined by one. Therefore, for evaluating password strength in the experiment, the neural network implemented using the PyTorch library [14] is also used as a method alternative to classical machine learning algorithms. The network consists of the following layers:

1) embedding layer, used for converting a password consisting of characters into a vector of numeric values;
2) long short-term memory (LSTM) layer, representing a special kind of recurrent neural network architecture, capable of learning long-term dependencies (which is important when working with long-length passwords [15]);
3) linear layer used for converting the internal state of LSTM layer into category scores.

The results of comparing the performance of the boosting algorithm and recurrent LSTM network on the test set are shown in Table 3.

The source code of the experiment can be found on the website[10].

[9] A simplified representation of the text as a bag (multiset) of its words without any regard to grammar or word order but with retention of information about their number.

[10] https://github.com/james116blue/password_strength_verifier. Accessed February 01, 2023.

It should be noted that the approach used in the experiment in no way reduces the hacker's capabilities in the case of phishing-, keylogger-, and man-in-the-middle attacks. However, according to the 2020 Data Breach Investigation Report [16], 89% of all hackings involve some type of credential abuse (brute force attacks and their subtypes, as well as attacks aimed at reusing credentials). This fact suggests that the approach to password strength using machine learning minimizes the risks for most attack vectors, thus also explaining a significant amount of current research in this area [17–21].

Thus, it may be concluded that the recurrent LSTM network outperforms conventional machine learning methods, bringing the password strength classification quality score closer to one.

CONCLUSIONS

The present work, which proposes an approach to building a password strength verifier using machine learning methods, compares several algorithms on a set of password data tagged by strength level. This approach is shown to have a number of advantages over classical password strength verification methods operating without machine learning techniques. Thus, using machine learning methods for verifier construction allows the formulation of password strength requirements in an arbitrary complex form, which relies on incidents only. In addition, the proposed approach allows better resistance to attacks comprising a subset of brute force attacks, as well as rainbow table attacks. In the first case, this is achieved due to the fact that a complex password whose strength is evaluated using the machine learning algorithm makes brute force attack impossible due to the huge time required for the attacker to accomplish this task. In the second case, attacking a complex password would only be possible by applying a large rainbow table, which would require, in turn, the use of a significant amount of attacker resources, thus making the risks of such an attack almost trivial.

Among the considered machine learning methods, recurrent neural networks demonstrate a particular efficiency. Here, the representation learning of the text password as a vector of numerical values, consisting in finding the embedding layer weights, occurs simultaneously with training of the entire network to maximize classification accuracy. This allows the neural network itself to choose such a password vectorization that is effective exactly for the task at hand. In addition, neural networks work with a sequence of characters, not just their presence in the password, thus allowing avoiding the simplification used in the "bag of words" approach. This allows the recurrent neural network to be recommended as one of the most promising research areas for building a password strength verifier.

Authors' contribution. All authors equally contributed to the research work.

REFERENCES

1. Conklin A., Dietrich G., Walz D. Password-based authentication: a system perspective. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. 2004; IEEE. https://doi.org/10.1109/HICSS.2004.1265412
2. Dell'Amico M., Michiardi P., Roudier Y. Password strength: An empirical analysis. In: *2010 Proceedings IEEE INFOCOM.* 2010; IEEE. https://doi.org/10.1109/INFCOM.2010.5461951
3. Chakrabarti S., Singhal M. Password-based authentication: Preventing dictionary attacks. *Computer*. 2007;40(6): 68–74. https://doi.org/10.1109/MC.2007.216
4. Shay R., Komanduri S., Kelley P.G., Leon P.G., Mazurek M.L., Bauer L., Christin N., Cranor L.F. Encountering stronger password requirements: user attitudes and behaviors. In: *Proceedings of the Sixth Symposium on Usable Privacy and Security.* 2010; Article 2. https://doi.org/10.1145/1837110.1837113
5. Selifanov V.V. Evaluation of the efficiency of the information protection system of state information systems from unauthorized access. In: *Integration of Science, Society, production and Industry: Collection of Articles of the International Scientific and Practical Conference*. 2016. P. 109–113 (in Russ.).
6. Ferreira J.F., Johnson S.A., Mendes A., Brooke P.J. Certified password quality: a case study using Coq and Linux pluggable authentication modules. In: *Integrated Formal Methods. IFM 2017. Lecture Notes in Computer Science.* V. 10510. Springer International Publishing; 2017. P. 407–421. https://doi.org/10.1007/978-3-319-66845-1_27
7. Alshare K.A., Lane P.L., Lane M.R. Information security policy compliance: a higher education case study. *Information & Computer Security*. 2018;26(1):91–108. https://doi.org/10.1108/ICS-09-2016-0073
8. AlSabah M., Oligeri G., Riley R. Your culture is in your password: An analysis of a demographically-diverse password dataset. *Computers & Security*. 2018;77: 427–441. https://doi.org/10.1016/j.cose.2018.03.014
9. Ji S., Yang S., Wang T., Liu C., Lee W.H., Beyah R. Pars: A uniform and open-source password analysis and research system. In: *ACSAC' 15*: *Proceedings of the 31st Annual Computer Security Applications Conference.* 2015. P. 321–330. https://doi.org/10.1145/2818000.2818018
10. Aizawa A. An information-theoretic perspective of TF–IDF measures. *Information Processing & Management*. 2003;39(1):45–65. https://doi.org/10.1016/S0306-4573(02)00021-3

11. Bishop C.M. *Pattern Recognition and Machine Learning*. New York: Springer; 2006. 738 p.
12. Lever J., Krzywinski M., Altman N. Classification evaluation: It is important to understand both what a classification metric expresses and what it hides. *Nat. Methods*. 2016;13(8):603–604. https://doi.org/10.1038/nmeth.3945
13. Medsker L.R., Jain L.C. (Eds.). *Recurrent Neural Networks*. *Design and Applications*. CRC Press; 2001. P. 64–67.
14. Imambi S., Prakash K.B., Kanagachidambaresan G.R. PyTorch. In: Prakash K.B., Kanagachidambaresan G.R. (Eds.). *Programming with TensorFlow*. EAI/Springer Innovations in Communication and Computing (book series). Springer; 2021. P. 87–104. https://doi.org/10.1007/978-3-030-57077-4_10
15. Yu Y., Si X., Hu C., Zhang J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput*. 2019;31(7):1235–1270. https://doi.org/10.1162/neco_a_01199
16. Jartelius M. The 2020 Data Breach Investigations Report–a CSO's perspective. *Network Security*. 2020;2020(7):9–12. https://doi.org/10.1016/S1353-4858(20)30079-9
17. Sarkar S., Nandan M. Password Strength Analysis and its Classification by Applying Machine Learning Based Techniques. In: *2022 Second International Conference on Computer Science, Engineering and Applications* (*ICCSEA*). IEEE, 2022. P. 1–5. https://doi.org/10.1109/ICCSEA54677.2022.9936117
18. Sakya S.S., Mauparna M.N. Building a Multi-class Password Strength Generator and Classifier Model by Augmenting Supervised Machine Learning Techniques. Preprint. 2022. https://doi.org/10.21203/rs.3.rs-1820885/v1
19. Murmu S., Kasyap H., Tripathy S. PassMon: A Technique for Password Generation and Strength Estimation. *J. Network Syst. Manage*. 2022;30(1):13. https://doi.org/10.1007/s10922-021-09620-w
20. Tran L., Nguyen T., Seo C., Kim H., Choi D. *A Survey on Password Guessing*. *arXiv preprint arXiv*:2212.08796. 2022. https://doi.org/10.48550/arXiv.2212.08796
21. Xiao Y., Zeng J. Dynamically generate password policy via Zipf distribution. *IEEE Transactions on Information Forensics and Security*. 2022;17:835–848. https://doi.org/10.1109/TIFS.2022.3152357

About the authors

Vladimir V. Belikov, Cand. Sci. (Military), Assistant Professor, Department of Information Security, Institute of Artificial Intelligence, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: belikov_v@mirea.ru. Scopus Author ID 57983605100, https://orcid.org/0000-0003-1423-1072

Ivan A. Prokuronov, Cryptographic Analysis Specialist, SFB Laboratory (56/2, Mishina ul., Moscow, 127083 Russia). E-mail: miltumultik@gmail.com. https://orcid.org/0000-0003-0999-311X

Об авторах

Беликов Владимир Вячеславович, к.воен.н., доцент, доцент кафедры информационной безопасности № 252 Института искусственного интеллекта ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: belikov_v@mirea.ru. Scopus Author ID 57983605100, https://orcid.org/0000-0003-1423-1072

Прокуронов Иван Андреевич, специалист инженерно-криптографического анализа, ООО «СФБ Лаборатория» (127083, Россия, Москва, ул. Мишина, д. 56, стр. 2). E-mail: miltumultik@gmail.com. https://orcid.org/0000-0003-0999-311X

Translated from Russian into English by Kirill V. Nazarov
Edited for English language and spelling by Thomas A. Beavitt