MARKOV, PROBABILISTIC AND RULE-BASED PASSWORD GUESSING METHODS: SURVEY AND COMPARISON

Internal PhD student, **Andrius Chaževskas**,
Vilnius University,
Akademijos st. 4, 08412 Vilnius, Lithuania,
Forensic science centre of Lithuania,
Lvivo st. 19A, -09313 Vilnius, Lithuania,
<andrius.chazevskas@mif.stud.vu.lt>

Prof., dr., **Igoris Belovas**,
Vilnius University,
Akademijos st. 4, 08412 Vilnius, Lithuania,
<igoris.belovas@mif.vu.lt>

Prof., dr., **Virginijus Marcinkevičius**,
Vilnius University,
Akademijos st. 4, 08412 Vilnius, Lithuania,
<virginijus.marcinkevicius@mif.vu.lt>

Summary

Offline password guessing is an important procedure for forensic encrypted data examination where the data must be decrypted first. The most common password guessing attacks are dictionary and brute-force, but the main drawback of a brute-force attack is the size of a set of all possible password candidates, which grows exponentially with the length of the password. The analysis of leaked password databases shows that users tend to use easy-to-remember passwords. It means that many passwords usually consist of a logical structure - they are not just random character sets. Forensic information technology experts could exploit this defect using different offline password guessing strategies relying on new password generation rules, machine learning, and natural language processing. This research offers a survey and comparison of the state-of-the-art password guessing methods such as Rule-based, Markov, Probabilistic Context-Free Grammar which can be applied in forensic IT examinations.

Keywords: offline password guessing, forensic examination, leaked database.

Introduction

The forensic information technology (IT) experts perform examinations of digital information following the tasks assigned by the courts and pre-trial investigation institutions. With the growing number of such examinations, the number of new cases related to encrypted information is also increasing. This problem directly affects the timing and quality of IT examinations. Working with the cases where the encrypted data must be decrypted first, using the password guessing procedures, the most important are the technical characteristics of the hardware used by laboratories for password guessing and the strategies of password guessing attacks selected by forensic experts. A password guessing attack is an attack that attempts to disclose a user's password by systematically trying possible passwords. The most common password guessing attacks are dictionary and brute-force. A brute force attack is an attack when all possible passwords from a defined set of characters are tested until the correct one is found. A dictionary attack is based on trying all the words in a pre-arranged dictionary. The laboratories can evaluate the possibility of applying the brute force attack using the formula (1):

$$MaxTime = \frac{A^M}{N} \qquad (1)$$

where A stands for the alphabet size, M stands for the password length, and N denotes the number of password guess attempts per second. It is obvious that the brute force attack is limited and depends on available hardware resources - how many guesses attempts can be made per second.

Our own[1] and others' studies[2,3], show that users tend to set their personal passwords predictably, favoring short strings, names, places, animals, ordinary and diminutive words typical of English and national language dictionaries. Figure 1 shows one hundred most common Lithuanian passwords (leaked from one of the Lithuanian social service providers) classified by different compositions regarding their structures and meaning. This information could be used to develop various password guessing techniques based on state-of-the-art password guessing methods, which generate dictionaries of new password candidates. This study introduces the main password guessing methods revealing the results obtained trying to guess passwords leaked from one of the Lithuanian social websites.

1 Chaževskas, A., Belovas, I., Marcinkevičius, V. (2021). Forensic password examination in leaked user databases. *Zborník príspevkov 17. medzinárodný kongres kriminalistika a forenzné vedy: veda, vzdelávanie, prax,* P. 241-257).

2 Bonneau, J. (2012). The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. *Proceedings - IEEE Symposium on Security and Privacy*, no. Section VII, P. 538–552: <https://doi.org/10.1109/SP.2012.49>.

3 Keika, Mori et al. (2020). Comparative Analysis of Three Language Spheres: Are Linguistic and Cultural Differences Reflected in Password Selection Habits? *IEICE Transactions on Information and Systems:* <https://doi.org/10.1587/transinf.2019ICP0009>.

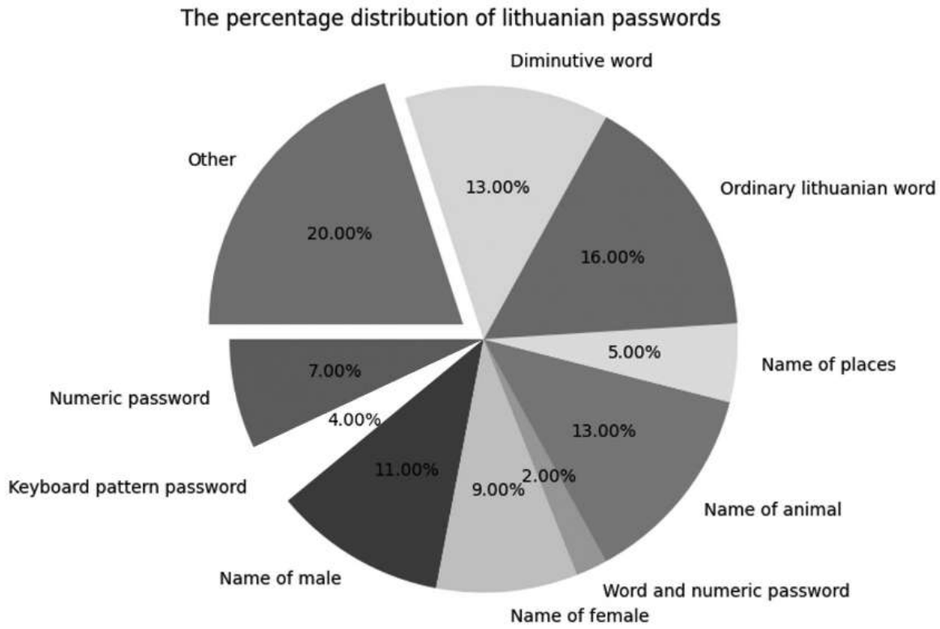

Figure 1. The percentage distribution of Lithuanian passwords.

The rest of this paper is structured as follows: Section 2 introduces related works and explains our research methods; In Section 3, we present our methodology, the password guesser and password guessing methods, evaluation metric, leaked password dataset, and dictionaries; Section 4 introduces results of performed experiments; Section 5 summarizes our results and presents the conclusions.

1. State-of-the-art password guessing methods: the literature survey

The offline password guessing attack is an attack that attempts to disclose a user's password by systematically trying possible passwords. Compared to brute-force attacks, different password guessing methods based on the creating the lists of new password candidates can construct dictionaries that are more similar to human password creation behavior, making them a more promising choice for offline password guessing strategy. This chapter presents state-of-the-art password guessing methods, which can be classified into three main categories: Rule-based, statistical (Markov, Probabilistic Context-Free Grammar), and Neural network models.

1.1. Rule-based method

The Rule-based method is a popular strategy in the digital forensics field and is widely used by forensic IT experts. Attacks of this type rely on password creation and reuse patterns extracted from previously leaked datasets or user surveys[4]. Special password guessing applications include *Passware Kit Forensic*, *Hashcat*, *John the Ripper*[5], and others. The main idea of the Rule-based method is combining various training lists (dictionaries) with mangling rules and

4 Nosenko, A., Cheng, Y., Chen, H. (2022). Learning Password Modification Patterns with Recurrent Neural Networks. *Secure Knowledge Management In The Artificial Intelligence Era.* 1549, P. 110–29: <https://doi.org/10.1007/978-3-030-97532-6_7>.

5 Password guessing software repositories: *Passware:* <https://www.passware.com>, *Hashcat:* <https://hashcat.net/hashcat/>, *John the Ripper:* <https://www.openwall.com/john/>.

transforming the original passwords (dictionary words) into new candidate passwords. Typical rules include appending characters, reversing the items, capitalizing the first letter, etc. Table 1 shows some basic rules[6] that can be created and applied with the *Hashcat* application.

Table 1. Examples of rules for password guessing.

Name	Function	Description	Example Rule	Input Word	Output Word
Nothing	:	Do nothing (passthrough)	:	p@ssW0rd	p@ssW0rd
Lowercase	l	Lowercase all letters	l	p@ssW0rd	p@ssw0rd
Uppercase	u	Uppercase all letters	u	p@ssW0rd	P@SSW0RD
Capitalize	c	Capitalize the first letter and lower the rest	c	p@ssW0rd	P@ssw0rd
Invert Capitalize	C	Lowercase first found character, uppercase the rest	C	p@ssW0rd	p@SSW0RD
Reverse	r	Reverse the entire word	r	p@ssW0rd	dr0Wss@p
Duplicate	d	Duplicate entire word	d	p@ssW0rd	p@ssW0rdp@ssW0rd
Reflect	f	Duplicate word reversed	f	p@ssW0rd	p@ssW0rddr0Wss@p
Rotate Left	{	Rotate the word left.	{	p@ssW0rd	@ssW0rdp
Rotate Right	}	Rotate the word right	}	p@ssW0rd	dp@ssW0r
Append Character	$X	Append character X to end	$1$2	p@ssW0rd	p@ssW0rd12
Prepend Character	^X	Prepend character X to front	^2^1	p@ssW0rd	12p@ssW0rd
Truncate left	[	Delete the first character	[	p@ssW0rd	@ssW0rd
Truncate right	]	Delete the last character	]	p@ssW0rd	p@ssW0r
Delete @ N	DN	Delete character at position N	D3	p@ssW0rd	p@sW0rd
Insert @ N	iNX	Insert character X at position N	i4!	p@ssW0rd	p@ss!W0rd
Overwrite @ N	oNX	Overwrite character at position N with X	o3$	p@ssW0rd	p@s$W0rd
Truncate @ N	,N	Truncate word at position N	,6	p@ssW0rd	p@ssW0
Replace	sXY	Replace all instances of X with Y	ss$	p@ssW0rd	p@$$W0rd

The table above shows that the Rule-based attack is flexible, accurate, and designed for fast password candidate generation. For example, having the Lithuanian dictionary, which consists of 83258 words, and using the rule which appends digits (from 0 to 9) to the end of every word, we will have a ten times larger dictionary. Besides, in our previous study (see the first footnote), the investigation of leaked Lithuanian passwords showed that all user-created different passwords had 3703 patterns of printable characters from the ASCII table. The revealed structures of patterns can help to create rules for effective password guessing. The ten most common patterns (see on the left side of thefigure) and the percentage distribution of all patterns with different lengths of character sets (see on the right side of the figure) are presented in Figure 2.

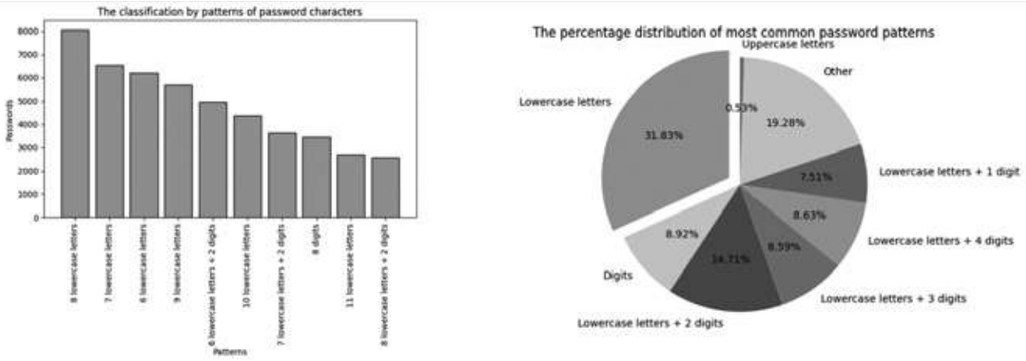

Figure 2. Most common password patterns.

6 *Hashcat Rule-based attack:* <https://hashcat.net/wiki/doku.php?id=rule_based_attack>.

It is not difficult to see that many user passwords (about 39 percent) contain various words with numbers at the end. The practical implementation of different rules using a *Hashcat* program is presented in Chapter 3. Also, the special password guessing programs can be used to implement a multi-stage hybrid password guessing attack that could employ different dictionaries, rules, and even fixed brute-force attacks in every new stage. Figure 3 shows possible password guessing attack settings by the *Passware* application.

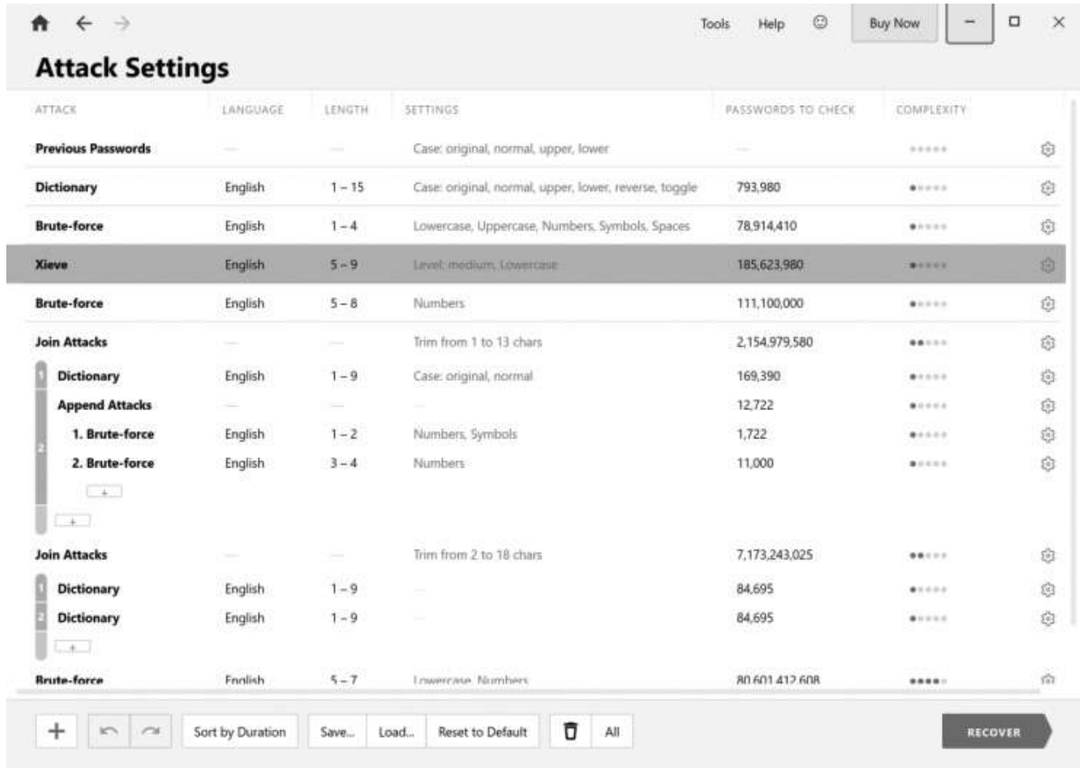

Figure 3. The password guessing attack settings of the Passware application.

1.2. The Markov method

A Markov chain is a model which describes a sequence of possible events, where each event depends on one or more of the most recent previous events. The Markov model was the dominant approach for modeling language, first introduced as a password guessing approach in 2005 by Narayanan et al.[7] The concept of the Markov model is to predict the next character based on the previous characters. In zero-order Markov's model, each character is generated according to the underlying probability distribution and independently of the previously generated characters. In the first-order Markov model, each character diagram (ordered pair) is assigned a probability. Each character is generated by looking at the previous (see formula (2)):

$$P\left(x_1, x_2 \ldots x_n \; QUOTE \; MaxTime = \frac{(A)^M}{N}\right) = v(x_1)\prod_{i=1}^{n-1} \quad v(x_{i+1}|x_i) \tag{2}$$

where P is the Markovian probability distribution on character sequences, x_i are individual characters, and the function is the frequency of individual letters and diagrams in English text. Later the Markov-based method, where users construct passwords by calculating the password's probability through the connection between characters from left to right,

7 Narayanan, A., Shmatikov, V. (2005). Fast dictionary attacks on passwords using timespace tradeoff. *Proceedings of the 12th ACM conference on Computer and communications security*. P. 364-372.

was used by Duermuth et al.[8] to develop a program known as Ordered Markov ENumerator (OMEN). This method is divided into two stages. During the training, the *n*-gram model is trained, and the frequency of each letter after a substring of length *n* is counted. During the generation stage, the probability of a probable password is calculated according to the Markov chain, and then the candidate passwords are generated. This method has three parameters: *n*-gram size, alphabet size, and the number of levels for enumerating passwords. The ***n*-gram size** parameter has the most significant impact on the method's accuracy. A larger *n* generally gives better results since larger *n*-grams provide a more accurate approximation to the password distribution. Still, also it implies a larger runtime, memory, and storage requirements. The larger size of an **alphabet** means that more parameters need to be estimated and that the runtime and memory requirements increase. However, a small alphabet implies that not all passwords can be generated. **A number of levels** is a third important parameter used to enumerate password candidates. As for previous parameters, a higher number of levels can potentially increase accuracy but also increase runtime. The implementation of OMEN and the results, trying to guess passwords from Lithuanian social website leaked data, are presented in Chapter 3 of this paper.

1.3. PCFG method

Probabilistic context-free grammar (PCFG) password guessing was introduced by Weir et al.[9] in 2009. PCFG-based methods analyze the password structures as grammars and divide passwords into different segment types according to their character composition. In this password guessing strategy, using password examples, you can create grammar rules that are used to generate new passwords. When parsing the training set, the algorithm denotes alphabet symbols as *L*, digit symbols as *D*, and special characters as *S*. For example, the password "**#password321**" would define with the simple structure $S_1L_8D_3$. Using sequences of already existing passwords, their segments are classified with the probability of each segment in the password. Using rewritten rules of the grammar, one can not only generate all passwords from the original dictionary but produce many new password candidates corresponding to password creation patterns learnt from the dictionary. In Table 2, we see a created rule that describes the two passwords "pass!word" and "kill@kill" in the sequence *S*.

Table 2. Example of probabilistic context-free grammar.

From left	to	right	Probability
S	->	$L_4S_1L_4$	1
L_4	->	pass	0,25
L_4	->	word	0,25
L_4	->	kill	0,5
S_1	->	!	0,5
S_1	->	@	0,5

In later research, semantic patterns were viewed as segments inserted into the dictionary to improve the efficiency of PCFG[10, 11]. Houshmand[12] et al. focused on enhancing the guessing performance for keyboard-walk structures and employed a smoothing technique. In their experiments, they achieved good performance on guessing keyboard-walk patterned passwords. Our results, obtained with the open-source PCFG version (developed by Matt Weir), are presented in Chapter 3.

1.4. Neural network-based methods

Neural network-based methods are used to construct password candidates. These methods can be divided into probabilistic that generate candidate passwords according to their probability and generative models that randomly generate candidates in batches. The Markov models utilize fixed-length context, while a long short-term memory (LSTM) network can store the features learned long ago. LSTM-based password guessing model denoted as FLA (Fast, Lean, and Accurate) was proposed by Melicher et al.[13] in 2016. FLA is similar to the Markov-based methods, for they both calculate the probability of a probable password by predicting the next character and its probability after fixed-length substrings. Neural networks can create language models in far less space than Markov models[14]. Figure 4 shows an example where a neural network tries to predict the next character of a password fragment. The network is being used to predict a 'd' given the context 'ba'. This network uses four characters of context. The probabilities of each next character are the output of the network. Post-processing on the network can infer probabilities of uppercase characters.

8 Dürmuth, M., Angelstorf, F., Castelluccia, C., Perito, D., Chaabane, A. (2015). OMEN: Faster password guessing using an ordered markov enumerator. In *Lecture Notes in Computer Science*. P. 119–132.

9 Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B. (2009). Password cracking using probabilistic context-free grammars. *Security and Privacy.* 30th IEEE Symposium. P. 391-405.

10 Veras, R., Collins, C., Thorpe, J. (2014). On the semantic patterns of passwords and their security impact. *In NDSS.*

11 Li, Z., Han, W., Xu, W. (2014). A large-scale empirical analysis of Chinese web passwords. *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14*). P. 559–574.

12 Houshmand, S., Aggarwal, S., Flood, R. (2015). Next Gen PCFG Password Cracking. *IEEE Trans. Inf. Forensics Secur*. 10, P. 1776–1791.

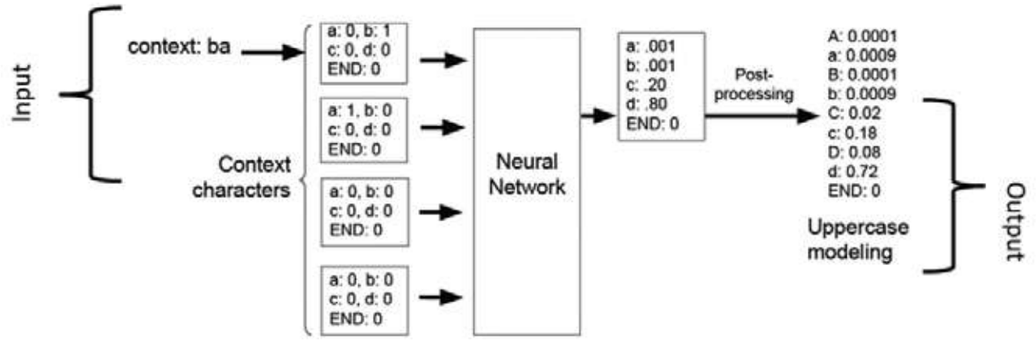

Figure 4. Neural network tries to predict the next character of a password fragment.

Liu et al.[15] combined the *PCFG* rule with the *LSTM* network called GENPass, which was applied to generate10^{12} password candidates and achieved significant improvement.

Generative adversarial network-based password guessing model *PassGAN*, introduced by Hitaj et al.[16], consists of one generating network which 'learns' and tries to find out the statistical structure of the data presented to it in order to create a new, statistically similar examples. The second network is used for testing (error detection), trying to detect samples generated by the first and the original (primary) networks. The training ends when the testing network can not distinguish the origin of the password. Both neural network-based methods need a longer time to train and generate password candidates. They also require a larger number of password candidates compared with other methods. Rule-based, *OMEN* and *PCFG* password guessing methods are advantageous in this respect.

2. Methodology

This section specifies leaked password datasets and the evaluation metric of the password guessing methods.

2.1. Guessing Data

The database dump file users.csv (leaked from one of the Lithuanian social websites) was downloaded from the Raid Forums[17] webpage. The file consists of 221445 rows and 80 columns. Since the main objects of this research are the users' passwords, only the unique passwords (114893 entries in total) were hashed with the SHA1 hash function and exported to the **HashSha1.txt** file using Python 3.7 script.

13 Melicher, W., Ur, B., Segreti, S. M., Komanduri, S., Bauer, L., Christin, N., Cranor, L. F. (2016), lean, and accurate: Modeling password guessability using neural networks. In: USENIX Security Symposium. P. 175-191.

14 Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S., Cernocky, J. (2012). *Subword language modeling with neural networks*: <http://www.fit.vutbr.cz/~imikolov/rnnlm/char.pdf>.

15 Liu, Y., Xia, Z., Yi, P., Yao, Y., Xie, T., Wang, W., Zhu, T. (2018). Genpass: A general deep learning model for password guessing with pcfg rules and adversarial generation. *2018 IEEE International Conference on Communications (ICC)*. P. 1-6.

16 Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F. (2019). A deep learning approach for password guessing. *International Conference on Applied Cryptography and Network Security*. P. 217-237.

17 Raid Forums is a database sharing and marketplace forum. There are exclusive database breaches and leaks plus an active marketplace: <https://raidforums.com/>.

2.2. Evaluation

The file HashSha1.txt with hashed passwords (114893 entries in total) was chosen as our target file to guess as many passwords as possible, applying different lists of passwords candidates generated with different dictionaries, Rule-based, OMEN, and PCFG password guessing methods. To compare different password guessing methods, we calculated the percentage of the password candidates (generated by password guessing methods using a training dataset) that were guessed (matched) in our target file. For password guessing, we have used *Hashcat* (version 6.2.5), which we employ to attack the hashed (SHA1) passwords (in HashSha1.txt file), trying different passwords candidate lists generated with our password guessing algorithms and rules.

2.3. Training data and hardware

In our experiments, we use Lithuanian[18] (83258 words in total) and English[19] (466550 words in total) dictionaries, as well as the dataset of leaked Lithuanian passwords[20] (523755 entries in total). For training and password guessing used desktop personal computer with main parameters: processor - Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz, 18 Cores; RAM - 128 GB; GPU - Nvidia GeForce RTX 3080 Ti.

2.4. Ethics

The study was conducted following the European Code of Conduct for Research Ethics prepared by ALLEA (the European Federation of Academies of Sciences and Humanities). Since the leaked databases consist of possible users' personal information (e.g., name, surname, email), all the information was securely stored and managed. Personal data exposure was prevented by implementing stringent security measures, and the passwords were not tested with real account services. This paper does not reveal any passwords.

3. Results

This section presents the results of the performed experiments with the different methods of password guessing.

3.1. Rule-based password guessing

During the experiments, we have tried different dictionaries and rule sets to increase the number of guessable passwords in the HashSha1.txt file with hashed passwords (114893 entries in total). The results of the experiments are summarized in Table 3. In the first stage, we use only the Lithuanian language dictionary *LitDict* (83258 words in total). From our previous study (see the first footnote), it has been known that the users are not inclined to employ specific Lithuanian letters (ą, č, ę, ė, į, š, ų, ū, ž), that is why at the second stage we replace them with the corresponding Latin letters (a, c, e, i, s, u, z), thus creating a new dictionary *LitDictAscii* (total 83258 words). This transformation allows us to guess more passwords (namely, 2320 with *LitDict* compared to 3517 with *LitDictAscii*). At the next stage, we complement our *LitDictAscii*dictionary by adding the English language dictionary *EnDict* (549808 words in the new merged dictionary). For this merged dictionary, we apply five simple rules (: - do nothing, r - reverse the entire word, u - uppercase all letters, l - lowercase all letters, c - capitalize the first letter and lower the rest), thus increasing our password guessability to 6.28% (see Table 3). Applying more rules (best64 rule set of the *Hashcat*), we get even better results - 16.28% guessed passwords. To improve the results, we have extended our dictionary by adding the dataset of leaked Lithuanian passwords (1073563 words in total). Applying different sets of rules (Best64, Specific, Rockyou-30000, Dive), we have achieved 69.72% password guessability of our hashed passwords list.

Table 3. Rule-based password guessing stages.

Stage	Password candidates	Guessed passwords	Guessed percent	Dictionaries and Rules
1	83258	2320	2.02 %	LitDict
2	83258	3517	3.06 %	LitDictAscii
3	549808	4614	4.02 %	LitDictAscii, EnDict
4	2748628	7221	6.28 %	LitDictAscii, EnDict, Basic rule set
5	42334773	19020	16.55 %	LitDictAscii, EnDict, Best64 rule set
6	82664027	46328	40.32 %	LitDictAscii, EnDict, Dataset, Best64 rule set
7	3864339263	56086	48.82 %	LitDictAscii, EnDict, Dataset, Best64, Specific rule sets
8	32206829929	75443	65.66 %	LitDictAscii, EnDict, Dataset, Rockyou-30000 rule set
9	106374865246	80104	69.72 %	LitDictAscii, EnDict, Dataset, Dive rule set

18 *Lithuanian dictionary*: <https://github.com/giekaton/lithuanian-words-txt>.
19 *English dictionary*: <https://github.com/dwyl/english-words>.
20 *Lithuanian passwords dataset*: <https://github.com/lexcor/LT-SecList>.

3.2. OMEN password guessing

The Ordered Markov ENumerator (OMEN) is the Markov model-based password cracker[21] that generates password candidates according to their occurrence probabilities. During the experiment, we generate new datasets with 10^5 and 10^6 password candidates, using our training dataset (password candidates of Lithuanian and English words and leaked Lithuanian passwords, 1073563 in total) and different *n*-gram size parameters, and calculate the percentage of matched passwords on the test dataset (see Table 4).

Table 4. The impact of n-gram size parameter.

n-gram	Password candidates	Guessed passwords	Password candidates	Guessed passwords
2	10^5	0.31 % (351)	10^6	1.10 % (1231)
3	10^5	0.47 % (541)	10^6	1.70 % (1953)
4	10^5	1.20 % (1382)	10^6	3.67 % (4214)
5	10^5	2.27 % (2605)	10^6	7.12 % (8186)

The results show that a larger *n*-gram usually provides a more accurate approximation of password distribution. The best results are obtained with *n*-gram size 5. Further research has been performed using this parameter and increasing the set of password candidates from 10^5 to $5{\cdot}10^9$. The summary of the experiment is presented in Table 5, where the number and the percentage of guessed passwords (using different amounts of password candidates) is provided.

Table 5. OMEN password guessing results.

Password candidates	Guessed passwords	Guessed percent
10^5	2605	2.27 %
10^6	8186	7.12 %
10^7	19254	16.76 %
10^8	34129	29.71 %
10^9	42674	37.14 %
$5{\cdot}10^9$	50537	43.99 %

3.3. PCFG password guessing

Further experiments have been performed using one of the probabilistic context-free grammars (*PCFG*) implementations[22]. The training dataset (password candidates of Lithuanian and English words and leaked Lithuanian passwords, 1073563 in total) has been trained with the *PCFG* trainer using the default parameters. We have generated new datasets from 10^5 to $5{\cdot}10^9$ password candidates and, using *Hashcat*, have performed the password guessing on our test dataset. The results are summarized in Table 6, which provides the number and the percentage of guessed passwords obtained using different amounts of password candidates.

Table 6. PCFG password guessing results.

Password candidates	Guessed passwords	Guessed percent
10^5	12109	10.54 %
10^6	25740	22.40 %
10^7	42633	37.11 %
10^8	51797	45.08 %

21 OMEN release: <https://github.com/RUB-SysSec/OMEN>.
22 PCFG cracker release: <https://github.com/lakiw/pcfg_cracker>.

10^9	57771	50.28 %
$5{\cdot}10^9$	60707	52.84 %

Trying to accurately compare all three methods (*OMEN*, *PCFG*, and Rule-based), we have generated new datasets from 10^5 to $5{\cdot}10^9$ password candidates using the *Rockyou*-30000 rule set (the default order of the rules has not changed). The results are summarized in Table 7, which provides the number and the percentage of guessed passwords obtained using different amounts of password candidates.

Table 7. Rockyou-30000 rule set password guessing results.

Password candidates	Guessed passwords	Guessed percent
10^5	112	0.10 %
10^6	536	0.47 %
10^7	2565	2.24 %
10^8	41274	35.92 %
10^9	58965	51.32 %
$5{\cdot}10^9$	67162	58.46 %

The result comparison of all applied methods (*OMEN*, *PCFG*, and *Rockyou*-30000) is given in Figure 5.

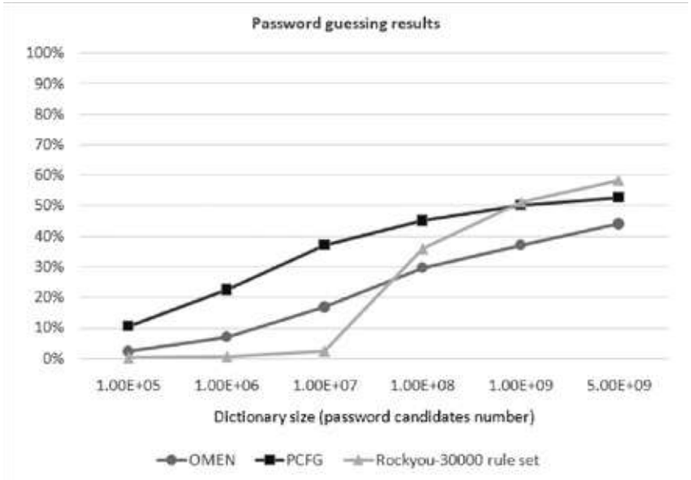

Figure 5. The results of password guessing.

Conclusions

The results of the experiments show that most of the passwords were guessed by the Rules-based method using the *Rockyou*-30000 rule set (58.46 % guessed passwords in total), followed by the probabilistic context-free grammar-based (*PCFG*) guesser (52.84 % guessed passwords in total). These methods have required us to generate $5{\cdot}10^9$ password candidates to guess more than half of our hashed passwords dataset.

The *PCFG* and the *OMEN* approaches have shown more accurate results while generating smaller numbers of password candidates. For example, the *PCFG* has guessed 37.11 % passwords, and the *OMEN* has guessed 16.76 % passwords, compared to the Rockyou-30000 rule set, which has guessed 2.24 % passwords using 10^7 possible password candidates. It shows that the *PCFG* and the *OMEN* methods could be used in practice to guess the passwords faster, and the Rule-based approach results depend heavily on the number of rules and the order in which they apply.

It should be emphasized that the proper selection of training data is crucial for the best results in password guessing. Using only Lithuanian and English dictionaries and applying rule-based (Best64 rule set) password guessing, we have guessed only 16.55 % of passwords, while adding a dataset of leaked passwords to our dictionaries, we have guessed 40.32 %.

MARKOVO TIKIMYBĖMIS IR TAISYKLĖMIS PAGRĮSTŲ SLAPTAŽODŽIŲ PARINKIMO METODŲ APŽVALGA IR PALYGINIMAS

Andrius Chaževskas,
Igoris Belovas,
Virginijus Marcinkevičius

Santrauka

Slaptažodžių nustatymas yra svarbi šifruotų skaitmeninių duomenų tyrimo procedūra, kurios metu teismo informacinių technologijų ekspertai, siekia iššifruoti vartotojų duomenis tolesniam jų tyrimui. Dažniausiai naudojami žodynų ir pilno perrinkimo slaptažodžių parinkimo metodai, tačiau pagrindinis pilno perrinkimo metodo trūkumas yra visų galimų slaptažodžių rinkinio dydis, kuris eksponentiškai auga didėjant slaptažodžio ilgiui. Nutekintų slaptažodžių duomenų bazių tyrimai rodo, kad vartotojai linkę naudoti lengvai įsimenamus slaptažodžius. Tai reiškia, kad daugelis slaptažodžių paprastai susideda iš loginės struktūros ir nėra tik atsitiktiniai simbolių rinkiniai. Teismo informacinių technologijų ekspertai gali išnaudoti šį trūkumą, naudodami skirtingus slaptažodžių parinkimo metodus, pagrįstus naujomis slaptažodžių kūrimo taisyklėmis, mašininiu mokymusi ir natūralios kalbos apdorojimu. Šiame darbe autoriai apžvelgia ir palygina Markovo grandinių, tikimybinių gramatikos taisyklių rinkinių ir naujų slaptažodžių kūrimo taisyklių taikymu grįstus slaptažodžių parinkimo metodus.

Raktiniai žodžiai: slaptažodžio parinkimas, ekspertinis tyrimas, nutekinta duomenų bazė.