









Article

Development of a Method for Determining Password Formation Rules Using Neural Networks

Leila Rzayeva ^{1,*} , Alissa Ryzhova ¹ , Merei Zhaparkhanova ¹ , Ali Myrzatay ¹ , Olzhas Konakbayev ¹ , Abilkair Imanberdi ¹ , Yussuf Ahmed ²  and Zhaksylyk Kozhakhmet ¹ 

¹ Research and Innovation Center “CyberTech”, Astana IT University, Astana 010000, Kazakhstan; 220756@astanait.edu.kz (A.R.); 220942@astanait.edu.kz (M.Z.); a.myrzatay@astanait.edu.kz (A.M.); o.konakbayev@astanait.edu.kz (O.K.); a.imanberdiyev@astanait.edu.kz (A.I.); zh.kozhakhmet@astanait.edu.kz (Z.K.)

² Department of Computing, Birmingham City University, Birmingham B4 7BD, UK

* Correspondence: l.rzayeva@astanait.edu.kz; Tel.: +7-777-533-9169

Abstract

According to the latest Verizon DBIR report, credential abuse, including password reuse and human factors in password creation, remains the leading attack vector. It was revealed that most users change their passwords only when they forget them, and 35% of respondents find mandatory password rotation policies inconvenient. These findings highlight the importance of combining technical solutions with user-focused education to strengthen password security. In this research, the “human factor in the creation of usernames and passwords” is considered a vulnerability, as identifying the patterns or rules used by users in password generation can significantly reduce the number of possible combinations that attackers need to try in order to gain access to personal data. The proposed method based on an LSTM model operates at a character level, detecting recurrent structures and generating generalized masks that reflect the most common components in password creation. Open datasets of 31,000 compromised passwords from real-world leaks were used to train the model and it achieved over 90% test accuracy without signs of overfitting. A new method of evaluating the individual password creation habits of users and automatically fetching context-rich keywords from a user’s public web and social media footprint via a keyword-extraction algorithm is developed, and this approach is incorporated into a web application that allows clients to locally fine-tune an LSTM model locally, run it through ONNX, and carry out all inference on-device, ensuring complete data confidentiality and adherence to privacy regulations.

Keywords: cybersecurity; password security; machine learning; neural networks; social engineering; digital forensic; cryptography; behavioral analysis



Academic Editors:
Pavlos Papadopoulos,
Nikolaos Pitropakis and
Sokratis Katsikas

Received: 23 June 2025

Revised: 20 July 2025

Accepted: 28 July 2025

Published: 31 July 2025

Citation: Rzayeva, L.; Ryzhova, A.; Zhaparkhanova, M.; Myrzatay, A.; Konakbayev, O.; Imanberdi, A.; Ahmed, Y.; Kozhakhmet, Z.

Development of a Method for Determining Password Formation Rules Using Neural Networks.

Information **2025**, *16*, 655. <https://doi.org/10.3390/info16080655>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the digital age, most people unknowingly expose personal details, such as names, birthdates, pet names, and hobbies through social media, which often form the basis of their passwords. These elements, especially when combined into common structural patterns or “masks,” can be easily exploited by attackers using social engineering. Although many tools exist to assess password strength based on traditional policies like length and character variety [1–3], they rarely account for the personalized logic behind how users form their passwords, as this would require historical login password mappings. This research focuses on the widespread tendency of users to ignore system-generated password suggestions,

with nearly 90% opting to create their own based on familiar and predictable patterns [4]. These patterns typically follow one of several common rules: dictionary words, numeric sequences (like phone numbers or birthdates), or symbolic masks—with symbols rarely used alone. The method introduced here, which evaluates the potential vulnerability of a password by analyzing open-source data and password manager exports, highlights how individual habits and online presence contribute to guessability. While additional authentication methods such as biometrics and two-factor verification are commonly used, standard passwords remain the pre-eminent method of access. As a result, they continue to be a major security risk: in 2024, over 5.5 billion accounts were compromised globally, nearly eight times more than the previous year [5]. Survey data further reveal that users often rely on familiar routines, with 42% mixing meaningful words and numbers, 34% following basic platform requirements, and 32% reusing parts of previous passwords [6]. Existing solutions are presented in the form of websites and, moreover, do not take into consideration specific password creation rules for a particular individual. Some solutions allow passwords like “namelastname1!” or do not analyze frequently recurring patterns in leaked password databases at all. The described approach focuses on the user’s personal habits, considers characteristic patterns, and offers a convenient format in the form of a Google Chrome extension with real-time analysis capability. The scientific contributions of this study include the following: a review of existing password cracking methods to identify recurring weaknesses in user-created passwords; behavioral analysis of how users construct passwords using insights derived from leaked password datasets; designing and training an LSTM-based (Long Short-Term Memory) neural network to identify recurring structural patterns in password composition; and an implementation of an interactive tool that helps users improve password security by providing personalized feedback with the use of social engineering.

This research differs from existing related works by focusing specifically on password breaches caused by the human factor, rather than purely technical weaknesses or brute-force vulnerabilities. While many prior studies analyze password strength or propose password checkers based on length, complexity, or entropy, our work highlights a different attack vector: in cases when an attacker gains access to the user’s previous login-password combinations (e.g., through local devices or leaked databases) and combines this with open data from the user’s social media accounts, they can deduce the user’s login/password generation rules, thereby making it substantially easier and faster for an attacker to gain unauthorized access to both previous and recent user accounts.

The novelty of our research lies in the proposed password validation method that accounts for the “human factor in the creation of usernames and passwords” and the factor of social engineering dependencies, which are overlooked in conventional checkers. By identifying these common substrings, we construct generalized password masks that highlight structural weaknesses, potentially enabling future compromise of similar accounts.

The structure of the paper is as follows: Section 2 provides an overview of neural network architectures and prior research on the influence of personal information in password creation. Section 3 presents evidence comprising real survey results of user’s credential formation and usage practices, describes a new method for weakness evaluation and recurrent user login/password creation rules, and presents proposals for processing all data directly on the user’s device using an ONNX model to ensure the privacy of personal data. Section 4 presents the experimental results and an analysis of the observed behavioral patterns. Section 5 concludes the paper and discusses the model’s potential practical applications.

2. Literature Review

Passwords were among the earliest safeguards in multi-user systems, yet research still shows a tension between security and usability. Early time-sharing hosts accepted simple dictionary words; administrators later tightened rules—minimum length, varied character classes, and so on—to curb rising breaches, for example by requiring at least eight symbols with digits and mixed case. Large-scale telemetry confirms that users keep trading entropy for memorability: a landmark web crawl found the median password short, predictable, and reused across sites [7]. Follow-up work identified four common constructions—dictionary word, numeric string, mask (“Aaaa11!”), and symbol-heavy variant [8]. Recent industry polling paints the same picture: 84% of people still embed names or dates in their credentials [4]. Bonneau’s comparative framework explains why text secrets persist despite decades of advice: they are the most deployable but also the least resilient factor [9].

Stricter composition policies failed to address the issue. In laboratory experiments, when users were forced to include capitals, numbers, and symbols, many just added “1!” or some other simple suffix, a phenomenon Komanduri termed minimum compliance [10]. This discovery, among others, prompted NIST to revise its 800-63B guideline in favor of extended pass-phrases and blacklist verifications rather than strict character regulations [11].

Attackers adapted just as quickly. Probabilistic context-free grammars (PCFGs) rank guesses by likelihood and break large leaks far faster than brute-force enumeration [12]. Neural models raise the bar again: an LSTM evaluator reaches more than 90% recall on public corpora while remaining “fast, lean, and accurate” [13]; PassGAN’s adversarial generator, for instance, produces human-like strings and attains up to 73% additional hits when combined with Hashcat [14]. In a specific context, extracting social media OSINT can lower the guess budget needed by a factor of ten [15].

Defensive tools have not entirely kept pace. Numerous strength meters consistently overrate entropy [16]; some even leak training secrets via inference attacks [17]. Server-side protection improves when operators tune Argon2 memory settings, yet field studies show many deployments accept vendor defaults, leaving roughly 40% more room for GPU-based cracking [18]. Client-side feedback helps—for example, Dropbox’s open-source zxcvbn combines dictionaries and heuristics to give instant, low-budget advice [19]—but adoption remains uneven. A four-month longitudinal study found that people who install password managers do craft longer, unique credentials, yet fewer than one-third keep the tools because of setup friction [20]. Similar usability trade-offs appear in mobile PIN research: even modern six-digit codes follow guessable patterns [21]. Personalized strength meters such as DPAR raise entropy without harming recall, but acceptance is still modest outside laboratory cohorts [22].

Meanwhile, major vendors advocate passkeys. Microsoft’s production rollout stores public/private key pairs in secure hardware that, for example, neutralise phishing by design [23]. Its 2023 Digital Defense Report claims that combining passkeys with MFA blocks more than 99% of credential-phishing attempts, yet also concedes that migration will be gradual because of legacy devices and limited user awareness [24]. The FIDO Alliance echoes this view: passwords and passkeys will coexist for years until ecosystems and habits align [25].

To summarise, the literature converges on four practical lessons. First, personal context inevitably leaks into passwords, so purely syntactic checks are insufficient. Second, rigid rules prompt superficial fixes unless paired with usable, meaningful feedback. Third, adaptive data-driven models—probabilistic or neural—best capture real guessability and guide users toward safer choices. Fourth, passwordless schemes promise long-term relief

but will not replace text secrets overnight, creating a window where enhanced evaluation and guidance for conventional passwords remain essential. These insights motivate the present study's LSTM-based evaluator, which blends statistical scoring with social-context filters and outputs measurable indicators such as average negative log-probability and leak incidence.

3. Materials and Methods

3.1. User Survey

To assess the scale of the problem under consideration, an anonymous online survey was conducted among residents of Kazakhstan regarding their password creation practices when registering on various online platforms.

The results of this survey confirmed previous statistical findings indicating that users tend to ignore recommended guidelines for creating strong passwords. In most cases, people rely on familiar and repetitive patterns when choosing their personal login and password combinations, aiming to simplify memorization and facilitate future logins and usage.

These findings highlighted the relevance of the issue and served as the starting point for this research.

The survey included 527 respondents categorized into the following age categories. The majority of participants were 25–34 years old (174 participants), and the other groups were 46+ (121), 18–24 (116), and 35–44 (116). Participants included students, educators, IT specialists, office workers, and others, allowing for a broad perspective on password-related habits and digital hygiene awareness. The survey aimed to capture real-world practices regarding password creation, memorization, and attitudes toward password management policies. Below are the key findings and their interpretation.

All participants took part in the study voluntarily and provided informed consent via a digital form prior to participation. The study did not collect any sensitive or personally identifiable information, and no demographic data were linked to password responses.

Respondents were selected by convenience sampling—the questionnaire was distributed voluntarily to students, teachers, IT professionals and office workers.

The survey was posted via Google Forms and distributed via social media (Telegram, Instagram) and university chat rooms.

The questionnaire included both closed (multiple choice) and open-ended questions, allowing participants to give their own wording. Main topics:

- Ways of creating passwords;
- How often passwords are updated;
- Attitudes towards security policies;
- Difficulties in memorization.

The purpose of the survey was to supplement the technical analysis with behavioral data and assess real user practices. This allowed better understanding of how weak passwords are formed and to link the identified vulnerabilities to personal login/password creation rules, which is critical for personalized analysis within the neural network model. Specifically, when a user requests a new-password quality assessment, the system performs personalized fine-tuning (locally) of the base model, leveraging the user's login and password historical data in combination with publicly available information retrieved from their social media profiles. This localized fine-tuning process ensures that the model adapts its evaluation to the user's unique behavioral patterns as logic, which could be called their "rules of login/password creation", and social engineering contextual background, thereby enhancing the accuracy and relevance of the password strength assessment.

One of the key questions focused on attitudes of toward periodic password updates, a common security requirement in organizational environments. The responses were diverse and revealed a gap between formal policy expectations and real-world user sentiment, as illustrated in Figure 1.



Figure 1. Survey results—perceptions of mandatory password updates.

The survey results illustrated in Figure 1 show that the perception of the need to regularly change passwords among users remains ambiguous. The largest share of respondents (34.2%) find this requirement annoying, while 32.9% recognize it as inconvenient but necessary. Only a quarter of respondents (25%) perceive updating passwords as an important security measure. Meanwhile, 7.9% of respondents do not think changing passwords is necessary at all. These data indicate that there is security fatigue and a need for more flexible, unobtrusive solutions in the area of credential management.

The data presented in Figure 2 suggest that password change habits are largely reactive rather than proactive. A significant 40% of respondents reported changing passwords only when they forget them, while another 25% do so approximately once a year. Regular updates are relatively rare: just 10% change passwords every few weeks, and 25% every few months. These results point to the fact that, for most users, password maintenance is not a routine security measure but a response to inconvenience or necessity.



Figure 2. Survey results—frequency of password changes for existing accounts (1—“About once a year”, 2—“Only when I forget a password”, 3—“Every few months”, 4—“Every few weeks”).

Figure 3 presents how respondents perceive the difficulty of remembering their passwords, using a scale from 1 (very easy) to 5 (very difficult). The most frequent response was level 3, selected by 185 participants, indicating a moderate level of difficulty. This suggests that while recalling passwords is not a major issue for most users, it is also not entirely effortless. The results reveal a range of user experiences: 66 respondents found it very easy (1), while 52 considered it very difficult (5). These extremes highlight the contrast between users who handle password recall with ease and those who find it challenging. Additionally, a notable number of users selected levels 2 (105) and 4 (119), which may imply that a portion of the population adopts insecure coping strategies such as reusing or writing down passwords.

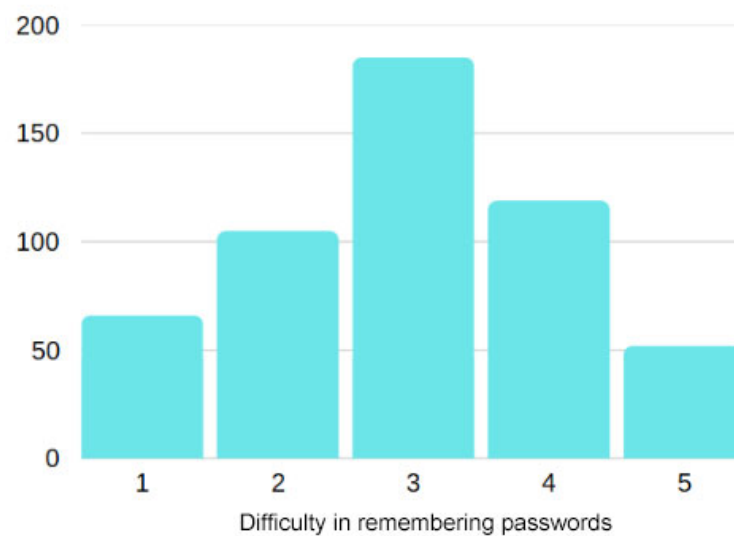


Figure 3. Survey results—difficulty in remembering passwords on a scale from 1 (very easy) to 5 (very difficult).

Taken together, these results underscore the importance of developing security solutions that go beyond static policy enforcement and instead account for real-world user behavior. While traditional password policies emphasize complexity and regular updates, the survey data reveal that many users perceive these measures as inconvenient, unnecessary, or difficult to follow in practice. This discrepancy suggests that rigid security requirements—without adequate consideration of usability—may lead to negative outcomes such as password reuse, oversimplification, or reliance on insecure storage practices (e.g., writing passwords down).

To address this, password management systems and security interfaces should incorporate adaptive mechanisms that respond to individual behavior and cognitive patterns. For example, systems could provide personalized recommendations based on users' password history or known habits, offering safer alternatives without overwhelming the user. Context-aware reminders triggered not by arbitrary time intervals but by risk signals or behavioral indicators—may encourage healthier password hygiene more effectively than fixed expiration policies.

3.2. New Detection Method of Recurrent Rules in Login/Password Creation

A new method, the logical explanation presented in Figure 4, is introduced by analyzing the transparency and predictability of logins and passwords.

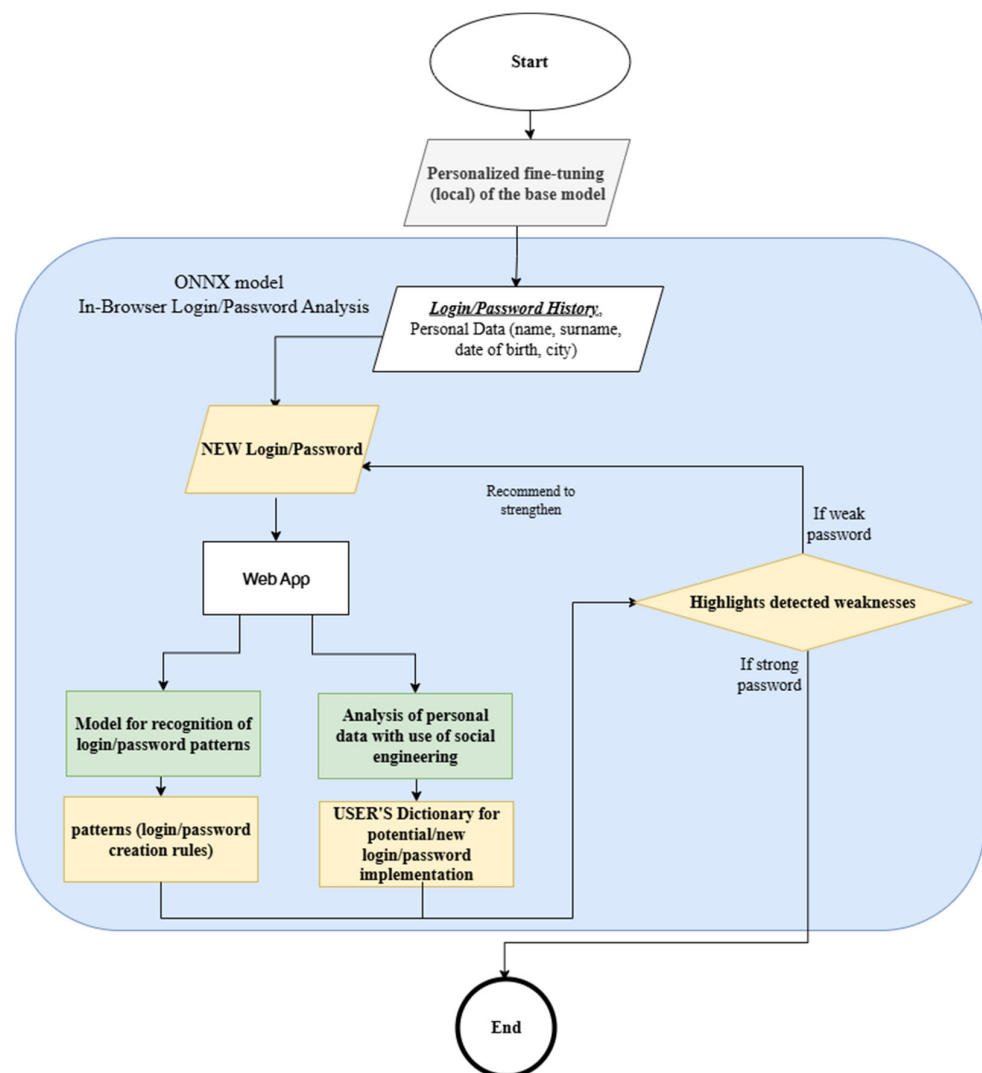


Figure 4. Logical flow-chart of proposed method.

It is important to highlight that this method is only applicable when the user's login and password history is available. This is precisely what makes the research both scientifically and practically novel. Existing methods do not take into account the fact that each user has their own creation rules.

The proposed method is a combination of two analytical components: first, pattern recognition in the structure of a user's login and password history; and second, identification of the user's personal "dictionary" by analyzing their social network activity. Based on the results of these two stages, the system evaluates the uniqueness and complexity of any newly created credential in real time and provides tailored recommendations regarding its strength.

To realise the research goal, a multi-level methodology was developed, including empirical data collection, technical processing of the data, model training and deployment of user solutions. The experimental material was based on a dataset generated from publicly available sources of compromised passwords (including RockYou 2024, LinkedIn v9.1.467, GitHub v3.17.2). After cleaning for duplicates, service characters and invalid strings, the final dataset contained more than 31,000 anonymised records suitable for sequential analysis.

Open-source datasets collected from Kaggle [26] and GitHub [27] were used as the training set. The DecisionTreeClassifier model was trained on a portion of these data,

with all tags being manually typed before training—each password was assigned a type. During training, the algorithm sequentially selects the most informative features and forms a hierarchical tree structure (Figure 5), where each internal node corresponds to a logical condition that allows the data to be divided into more homogeneous subgroups according to the target feature.

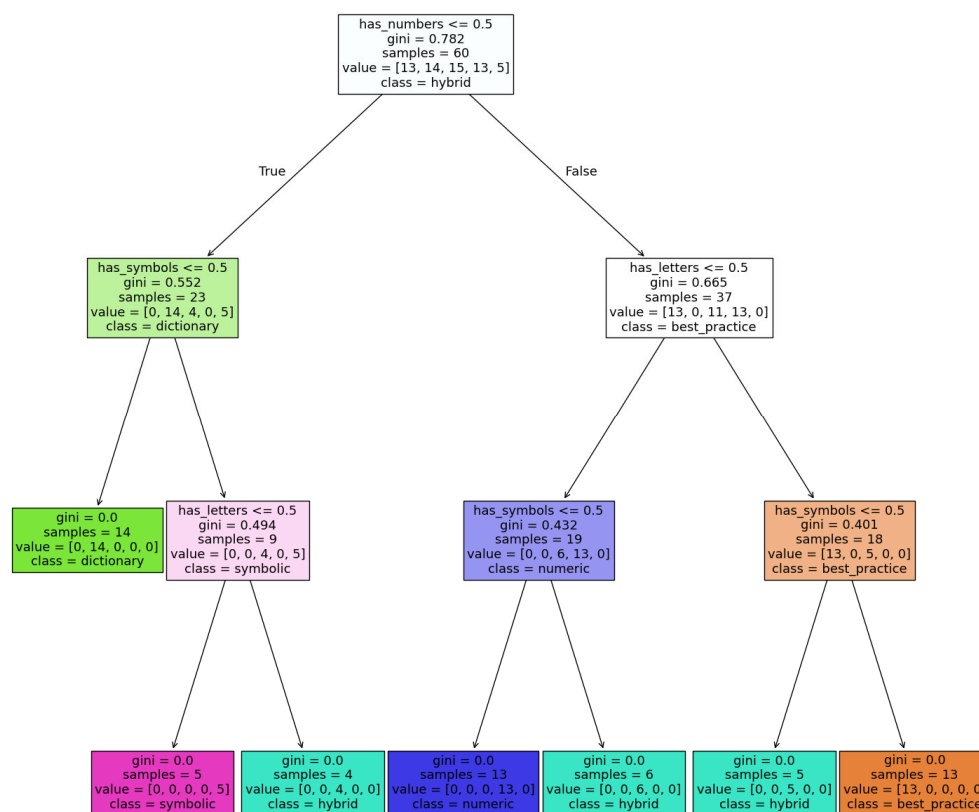


Figure 5. Decision tree results.

After training was completed, the model was tested on a deferred sample, which made it possible to evaluate its ability to generalize patterns to new, previously unencountered data. To analyze the results, model predictions were obtained and then compared with actual class labels, which made it possible to quantitatively assess the accuracy of the classification. However, this solution was subsequently transformed in favour of more flexible and expressive models capable of taking into account the sequential nature of passwords and identifying complex latent dependencies in the structure of characters.

The decision tree classifier learns a mapping:

$$f(x) \rightarrow \{0, 1, \dots, K - 1\} \quad (1)$$

where $x = [x_1, x_2, x_3]$ are binary features indicating the presence of letters (x_1), digits (x_2), and symbols (x_3) in a password, and K is the number of possible password classes.

At each node, the model selects a feature and threshold to split the data in a way that minimizes the Gini impurity:

$$G = 1 - \sum p_i^2 \quad (2)$$

where p_i is the proportion of class i within the node. The tree recursively partitions the feature space and assigns the most frequent class label at each leaf. This results in an interpretable model for password type classification based on lexical properties.

Decision trees were selected as one of the baseline models due to their widespread use in classification tasks. This algorithm was employed in a preliminary experiment to classify all passwords based on simple features such as the presence of letters, digits, symbols, and their combinations. This allowed the identification of typical structures and recurring patterns in passwords, which subsequently served as a foundation for deeper analysis and the construction of generalized masks. Thus, the decision tree model acted as an initial analytical stage from which hypotheses were formed, leading to the use of more flexible and expressive models such as LSTM.

Another investigated approach is a model based on Probabilistic Context-Free Grammar (PCFG) designed to identify stable structures in user passwords. The method involves splitting a password into logical fragments—sequences of letters (L), digits (D) and special characters (S)—and then constructing a probabilistic grammar based on the length and type of these segments.

For each password, its structure was generated, for example, L5-D4 for a password of the form Merei1611. From a formal standpoint, the PCFG model decomposes a password p into segments s_1, s_2, \dots, s_n , where each segment $s_i = (\tau_i, v_i)$, with $\tau_i \in \{L, D, S\}$ representing the type (Letter, Digit, or Symbol), and v_i the corresponding substring.

The structural pattern of a password is expressed as follows:

$$\sigma(p) = \tau_1|v_1| - \tau_2|v_2| - \dots - \tau_n|v_n| \quad (3)$$

For example, the password “Merei1611” corresponds to the structure L5–D4.

The probability of a structure is defined as follows:

$$P(\sigma) = C(\sigma)/N$$

where $C(\sigma)$ is the frequency of structure σ in the dataset, and N is the total number of observed structures.

The conditional probability of a segment is as follows:

$$P(v_i | \tau_i) = C_i(v_i)/\Sigma C_i \quad (4)$$

where $C_i(v_i)$ is the count of substring v_i among segments of type τ_i , and ΣC_i is the total number of such segments (Figure 6).

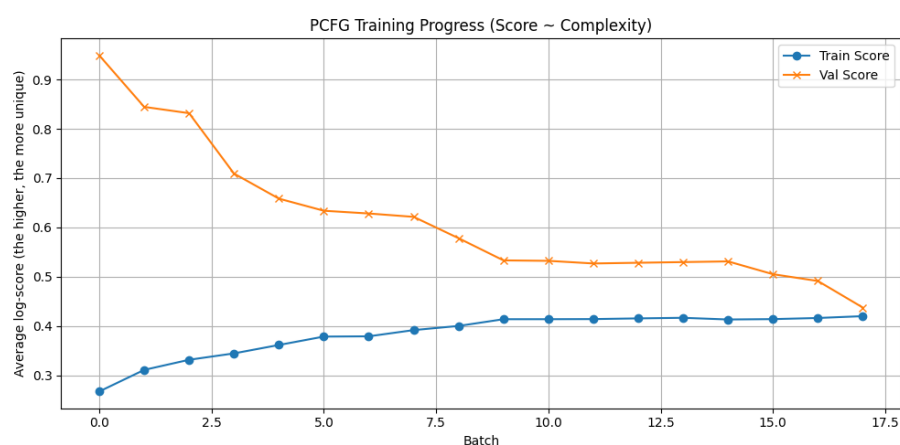


Figure 6. PCFG training progress.

Assuming independence between segments, the overall probability of the password is as follows:

$$P(p) = P(\sigma(p)) \times \prod P(v_i | \tau_i) \quad (5)$$

Password strength is then calculated as the normalized negative base-10 logarithm of this probability:

$$\text{Score}(p) = -(1/|p|) \times \log_{10} P(p) \quad (6)$$

A greater score signifies a rarer and more robust password. This approach enables the model to assess the distinctiveness and predictability of passwords by analyzing their structural and lexical patterns. The model underwent training on a tailored dataset, capturing the occurrence frequencies of these structures along with the frequencies of specific substrings within the categories. During the password assessment stage, the model employs a logarithmic metric that represents the likelihood of the structure and its elements adjusted for length.

The model was trained iteratively, with progress monitoring by average score and accuracy of recognizing known structures. Two graphs were created for visualization: one shows the change in the average logarithmic score on the training and validation samples as the training batch increases (Figure 7), and the other shows the accuracy of matching structures on both samples (Figure 6). This allows tracking of how well the model adapts to new data and captures recurring patterns in passwords.

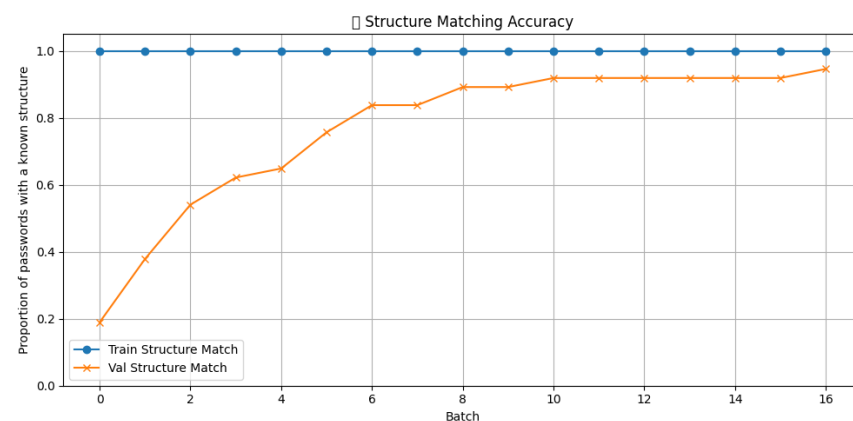


Figure 7. Structure matching accuracy.

It turns out that PCFG and decision tree models rely on predefined rules and characteristics for password analysis. PCFG represents passwords as structured segments and uses probabilistic estimation based on their frequency, while decision trees classify data according to explicitly defined characteristics, such as length or character types.

The PCFG model was chosen as a comparative baseline due to its established role in password structure analysis. In this approach, passwords are decomposed into logical segments (e.g., letters, digits, symbols), and probabilistic rules are generated based on the frequency and arrangement of these segments in the dataset. This method was applied to explore the distribution of structural patterns and to estimate the likelihood of specific password formats. The PCFG model enabled assessment of the strength and predictability of passwords using a rule-based probabilistic framework. While effective for capturing common patterns, its limitations in handling non-standard or user-specific behaviors motivated the transition to more adaptive models such as LSTM.

However, neither approach is flexible enough. They require manual feature specification and do not take into account the human factor in password creation. Decision trees use rigid logical divisions, and PCFG does not cope well with non-standard patterns, individual habits, or emotionally charged combinations.

These limitations led to a shift towards neural network models, such as LSTM, which are capable of learning from raw data and identifying complex, personalized patterns without manual rule configuration. The basis for password analysis in this paper is a neural

network model based on the LSTM architecture. This architecture was chosen due to its ability to efficiently process sequential data and capture temporal dependencies between elements of the input sequence. These properties are critical in the context of password analysis, where even minor permutations of characters can significantly alter the semantics and complexity of a password. Moreover, LSTM models have proven effective in tasks such as text generation, next-element prediction, and behavioral pattern recognition.

3.3. Technical Realization of the Proposed Method

The architecture of the developed model comprises three logical levels. The first stage involves converting input characters into dense vector representations of fixed dimensionality. This approach enables more efficient training than classical one-hot encoding, as it allows the model to capture syntactic and structural features of characters rather than merely their position in the alphabet.

At the second stage, a two-layer LSTM block is applied, which analyzes contextual relationships across the entire password string. This enables the model to recognize stable combinations of letters and digits, as well as positions of special characters characteristic of user habits. The use of two layers facilitates the processing of both short- and long-term dependencies between characters, while a dropout regularization mechanism helps prevent overfitting on limited datasets.

The final stage includes a fully connected layer that transforms the LSTM output states into a probability distribution over all possible characters in the vocabulary. This enables the model to both predict the next character and estimate the likelihood of specific structural patterns appearing in a password.

The primary distinction between the LSTM model proposed in this study and the neural network approach introduced by Melicher et al. [13] lies in the level of abstraction and the underlying objectives of modeling password structure. Melicher et al. employ a character-level recurrent neural network aimed at estimating the probability of individual character transitions, thereby enabling fine-grained modeling of password guessability based on positional dependencies. Their architecture is optimized for client-side password strength estimation and large-scale adversarial simulation, focusing on the statistical likelihood of character sequences within passwords.

In contrast, the work presented here shifts the emphasis toward the behavioral dimension of password generation, aiming to uncover structural regularities and semantically meaningful patterns that reflect human cognitive habits and social influences. The proposed LSTM architecture is designed to detect recurrent substrings and typical combinations such as names, and birth years with the goal of constructing generalized structural masks. These masks abstract the compositional logic employed by users, allowing the model to infer common behavioral tendencies that shape password formation rules.

Thus, while Melicher et al. make a significant contribution to password guessability modeling from a probabilistic standpoint, our approach focuses on the socio-technical aspect of password creation, offering complementary insights into the human factors that contribute to predictable and vulnerable password structures.

Training was performed using the Adam optimization algorithm, which ensures fast and stable convergence in deep neural networks. Cross-entropy was used as the loss function, which is standard for multi-class classification problems. To ensure result stability and reproducibility, a fixed seed value was set across all modules involving randomness. This allows the experiments to be repeated under identical conditions, yielding consistent results. The process was conducted on a local workstation equipped with a graphics accelerator, enabling reasonable training times even when processing large volumes of string data. The training dataset consisted of thousands of passwords sourced from public

leaks, which were pre-cleaned to remove duplicates and non-informative characters. Input sequences typically ranged in length from 8 to 24 characters, consistent with common user password practices.

The following formulas describe the LSTM operations for the first layer of the PasswordLSTM model, with embedding_dim = 128, hidden_dim = 256, and num_layers = 2, as per the PyTorch (2.6.0+cpu) torch.nn.LSTM documentation.

Input gate (i_t)—a fundamental component of the LSTM architecture—is responsible for regulating the inclusion of new information into the cell state. It controls the extent to which the input data at the current time step influences the memory cell update.

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \quad (7)$$

$$\text{where } W_{ii} \in R^{256 \times 128}, W_{hi} \in R^{256 \times 256}, b_{ii}, b_{hi} \in R^{256}, x_t \in R^{128}, h_{t-1} \in R^{256}$$

$$x_t \in R^{128} - \text{input vector at time step } t$$

$$h_{t-1} \in R^{256} - \text{hidden state from the previous time step}$$

$$W_{ii} \in R^{256 \times 128} - \text{weight matrix for input}$$

$$W_{hi} \in R^{256 \times 256} - \text{weight matrix for hidden state}$$

$$b_{ii}, b_{hi} \in R^{256} - \text{bias vectors}$$

$$\sigma(\cdot) - \text{sigmoid activation function}$$

Forget gate (f_t) is the part of the LSTM that decides what information to delete from memory. It receives current data and the previous state as input and outputs numbers from 0 to 1, where 0 means “forget it completely” and 1 means “leave it as it is”.

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \quad (8)$$

where

$$x_t \in R^{128} - \text{input vector at time step } t$$

$$h_{t-1} \in R^{256} - \text{hidden state from the previous time step}$$

$$W_{if} \in R^{256 \times 128} - \text{weight matrix for input } x_t$$

$$W_{hf} \in R^{256 \times 256} - \text{weight matrix for hidden state } h_{t-1}$$

$$b_{if}, b_{hf} \in R^{256} - \text{bias vectors}$$

$$\sigma(\cdot) - \text{sigmoid activation function}$$

Cell gate (candidate cell state) (C_t) is the part of LSTM that creates new content for memory, i.e., it suggests what can be added. It calculates possible new values using the input and previous state, but does not add them directly—this is done by the input gate.

$$C_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \quad (9)$$

where

$$x_t \in R^{128} - \text{input vector at time step } t$$

$$h_{t-1} \in R^{256} - \text{hidden state from the previous time step}$$

$$W_{ig} \in R^{256 \times 128} - \text{input weight matrix for candidate cell}$$

$$W_{hg} \in R^{256 \times 256} - \text{hidden weight matrix for candidate cell}$$

$$b_{ig}, b_{hg} \in R^{256} - \text{bias vectors}$$

$\tanh(\cdot)$ – hyperbolic tangent activation function

Output gate (o_t) is the part of the LSTM that decides which part of the current memory state to show to the outside, i.e., what will become the new hidden state h_t . IT uses the current memory C_t , passes it through \tanh , and filters the result using sigmoid (values from 0 to 1).

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \quad (10)$$

where

$$x_t \in R^{128} - \text{input vector at time step } t$$

$$h_{t-1} \in R^{256} - \text{hidden state from the previous time step}$$

$$W_{io} \in R^{256 \times 128} - \text{input weight matrix for output gate}$$

$$W_{ho} \in R^{256 \times 256} - \text{hidden weight matrix for output gate}$$

$$b_{io}, b_{ho} \in R^{256} - \text{bias vectors}$$

$\sigma(\cdot)$ – sigmoid activation function

Cell state (C_t) is the process of updating the internal state of memory C_t in the LSTM. It combines the old memory C_{t-1} , multiplied by the forget gate, and the new proposal from the cell gate \tilde{C}_t , multiplied by the input gate.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (11)$$

where

$$C_{t-1} \in R^{256} - \text{previous cell state}$$

$$f_t \in R^{256} - \text{forget gate output}$$

$$i_t \in R^{256} - \text{input gate output}$$

$$\tilde{C}_t \in R^{256} - \text{candidate cell state}$$

\odot denotes element – wise multiplication

Hidden state (h_t) is the output of the LSTM at the current step, what the LSTM “gives out”. It is calculated based on the updated memory C_t (passed through \tanh) and the output gate o_t , which decides what part of it to demonstrate.

$$h_t = o_t \odot \tanh(C_t) \quad (12)$$

where

$$C_t \in R^{256} - \text{current cell state}$$

$$o_t \in R^{256} - \text{output of output gate}$$

$$h_t \in R^{256} - \text{resulting hidden state}$$

$\tanh(\cdot)$ – hyperbolic tangent activation function

\odot denotes element – wise multiplication

In conclusion, the LSTM architecture was selected for its balanced combination of accuracy, computational efficiency, and interpretability. It not only detects repetitive struc-

tures and vulnerabilities in user passwords but also holds promise for generating secure yet memorable combinations, making it highly applicable in practical cybersecurity contexts.

Figure 8 illustrates the evolution of the loss function during the training of the PasswordLSTM model over 43 epochs. At the initial stage, both the training and validation loss curves show a sharp decrease, particularly within the first 10 epochs. This rapid drop indicates that the model quickly captures core structural dependencies in password sequences and effectively minimizes prediction error early in the learning process.

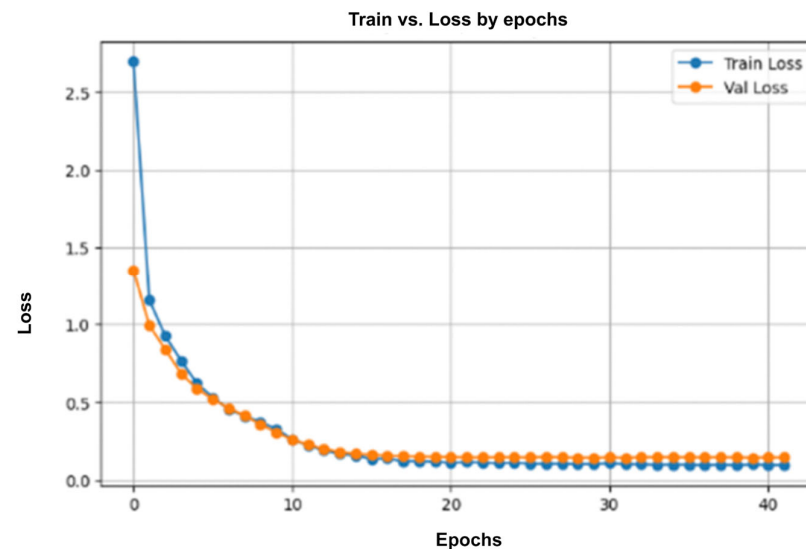


Figure 8. Training vs. validation loss.

After epoch 10, the rate of loss reduction gradually slows, and by approximately epoch 15, both curves stabilize at values below 0.15. This plateau suggests that the model has reached convergence, with only marginal improvements beyond that point. The close alignment between the two curves throughout the training process reflects a strong generalization ability and an absence of overfitting, which is particularly important when working with character-level sequence data.

Overall, the training loss dynamics confirm that the model was well-tuned and trained on a clean and representative dataset. The consistent behavior of both curves across epochs demonstrates the robustness of the LSTM architecture and the adequacy of the preprocessing steps, including character tokenization, sequence normalization, and feature enrichment. These results confirm the reliability of the model in password structure recognition tasks and its readiness for practical deployment.

Figure 9 presents the classification accuracy of the PasswordLSTM model across 43 training epochs on both training and validation datasets. A notable increase in accuracy is observed within the first 10 epochs, where the model transitions from near-random performance to a highly accurate state. This steep growth indicates the model's ability to rapidly extract meaningful patterns and semantic relationships within password sequences, despite the limited character set and relatively short input lengths.

Around epoch 15, both training and validation accuracy stabilize above 95%, with minimal divergence between the two curves. The close alignment of these metrics suggests that the model is not only accurate on seen data but also generalizes well to previously unseen samples. The consistent tracking of validation accuracy alongside training accuracy implies that the network did not overfit, even after prolonged exposure to the training set, which is often a challenge in character-level sequence models.

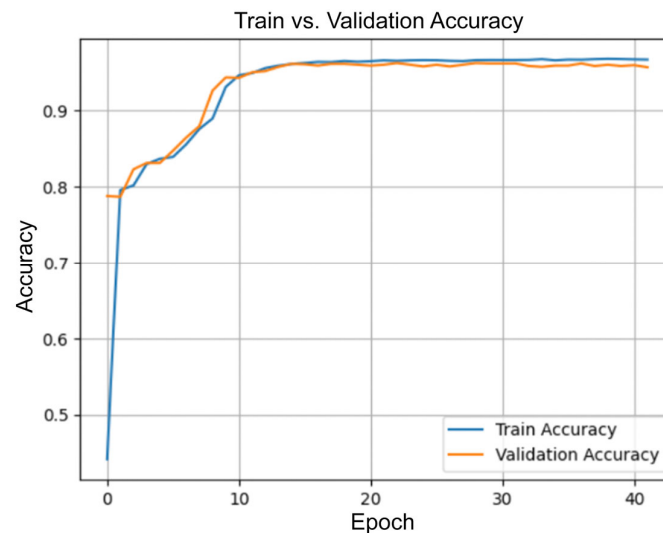


Figure 9. Training vs. validation accuracy.

The high and stable accuracy across later epochs also validates the quality of the data preprocessing pipeline and the effectiveness of the regularization techniques applied during training. Dropout between the LSTM layers and fixed sequence normalization likely contributed to the model's robustness. These results reinforce that the model is capable of maintaining high performance in realistic, user-facing scenarios, such as live password strength assessment or pattern analysis in browser-based security tools.

The training results presented in Figures 5 and 6 confirm the robustness and reliability of the proposed LSTM model. The rapid convergence of the loss function and the stable accuracy above 95% on both training and validation sets demonstrate that the model effectively captures underlying patterns in password data without overfitting. The close alignment of the curves indicates strong generalization, making the model well-suited for real-world applications involving password structure analysis and prediction. Overall, the training process was successful, resulting in a stable and high-performing architecture.

To evaluate the impact of publicly available personal data on the structure of user passwords, a two-stage analysis was performed. At the first stage, text snippets for queries containing name, city, date of birth, and address were collected using the Serper API. After text normalization, key tokens were matched to passwords from the corpus. Next, exact and partial name/login matches as well as semantic proximity to OSINT tokens were evaluated using RapidFuzz metrics. The results were combined with the main password attributes (length, presence of digits and symbols) and presented as a correlation matrix.

The analysis in Figure 10 showed that password length is positively correlated with its similarity to the name ($r \approx 0.65$), while partial matches are strongly correlated with both full similarity ($r \approx 0.87$) and semantic evaluation ($r \approx 0.42$). Meanwhile, adding digits or wildcards hardly reduces the predictability of a password if personal tokens remain in it.

Thus, names and related elements remain a key source of vulnerability: simply complicating the structure without removing such fragments does not provide reliable protection.

After the training phase, the model was exported in ONNX format, which allowed it to be integrated into two application tools: a web application developed in Vue 3, and a Google Chrome browser extension. This way, all of the processing is performed locally and without sending sensitive data to remote servers. By avoiding server-side storage or analysis, this design ensures maximum data confidentiality, as no user credentials are collected, stored, or processed on the server side.

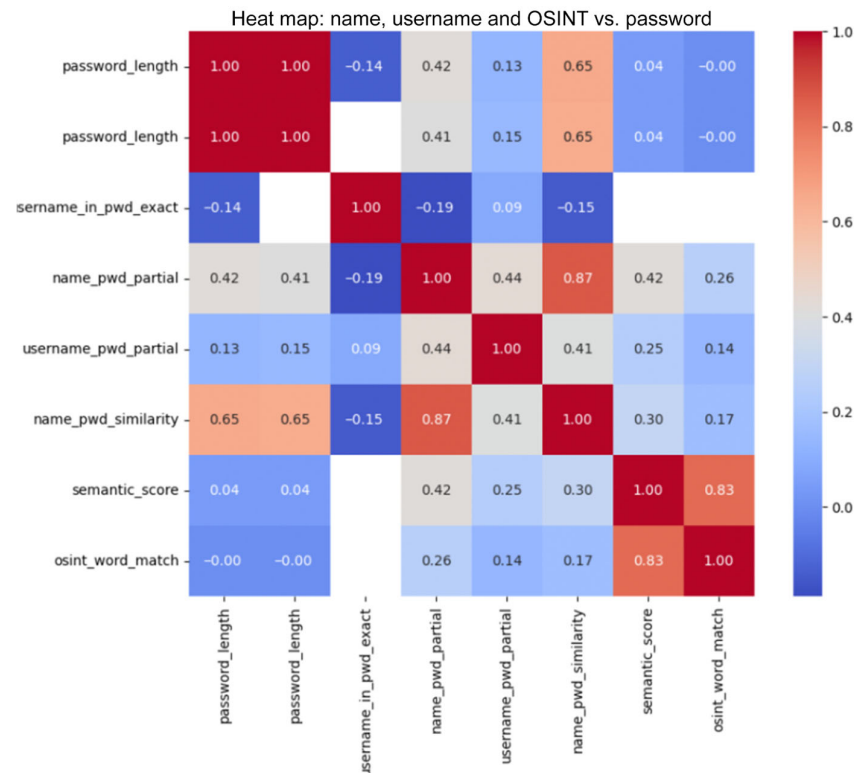


Figure 10. Heat map describing correlations between password and personal data.

The interfaces function in real time. After entering a password, the user receives instant feedback: visualization of the string decomposition into structural elements, assignment of the security level (WEAK, MED, STRONG), and probabilistic evaluation of its guessability based on the output distribution of the model. The extension uses IndexedDB to store repetitive patterns, which allows tracking typical input patterns and tailoring recommendations to the user's individual behavioral habits. Thus, the model implementation is not only focused on technical accuracy, but also on practical value in everyday cyber hygiene.

Building on the LSTM model's probabilistic scoring, the system was embedded into a comprehensive password-analysis platform that delivers real-time feedback through both a browser-based web app and a Chrome extension. In the web application, the ONNX (v1.18.0) -exported model and a character-to-index dictionary are fetched on page load via onnxruntime-web (1.22.0.) (WASM), ensuring all inference runs entirely client-side. As soon as a user types a password into the bound input field and clicks "Check," an asynchronous routine first validates that the field is not empty—resetting scores and emitting a console warning, if necessary, then calls the predictStrength function to compute two key metrics: the average negative log-probability (avg $(-\log P)$), and a normalized strength percentage (0–100). These values update reactive Vue.js variables, instantly repainting a colored label—red for WEAK (<40%), yellow for MED (40–70%), and green for STRONG ($\geq 70\%$)—alongside the raw avg metric for power users who want deeper insight (Figure 11).

Once scored, the system cross-references the password against a precomputed list of vulnerable substrings extracted during training, flagging any matches with warning icons to highlight structural weaknesses. To guard against credential-reuse attacks, it then sends only the first five characters of the password's SHA-1 hash to the Have I Been Pwned k-anonymity API (v3), retrieves the matching suffix list, and filters locally to display any breach count without ever exposing the full secret. Below the main panel, a dynamic password chart (Figure 12) renders the top seven most-reused passwords, and text lists

titled “Masks that make passwords easy to guess” and “Recommended to avoid” (Figure 13) guide users away from both generic and personalized patterns.

Password Strength Analyzer

WEAK

Data-breach check: ✗ Found 12,424,625 times!

avg (−log P): 1.937

Password's strength: 44 %

Figure 11. Web application: password analysis.



Figure 12. Web application: most reused passwords chart.

! Recommended to avoid

- 200
- 2004
- 004
- rkh
- hap
- apa
- par
- hapa
- apar
- hapar

Figure 13. Web application: patterns that are recommended to avoid.

The Chrome extension mirrors this logic on every site with `<input type="password">` fields. Defined in manifest.json, it comprises a popup UI, background worker, and content script built with Vue 3, TypeScript 5.8, and Vite 7.0. During onboarding, users optionally enter their name and city, triggering a DuckDuckGo lookup that scrapes meaningful keywords for personalized pattern checks and stores them locally in chrome.storage and IndexedDB. The content script then injects a live badge next to each password field and listens for keystrokes; on every input it runs the same ONNX inference, pattern matching, and breach lookup via a unified rate() function, updating the badge and inline diagnostics in milliseconds. Hot-reload support via chrome.runtime messaging keeps patterns up to date without refreshing pages. All processing—model execution, keyword parsing,

pattern checks—occurs on the user’s device, with only non-identifying hash prefixes or search queries leaving the browser, fully preserving privacy while delivering fast, insightful feedback (Figure 14).

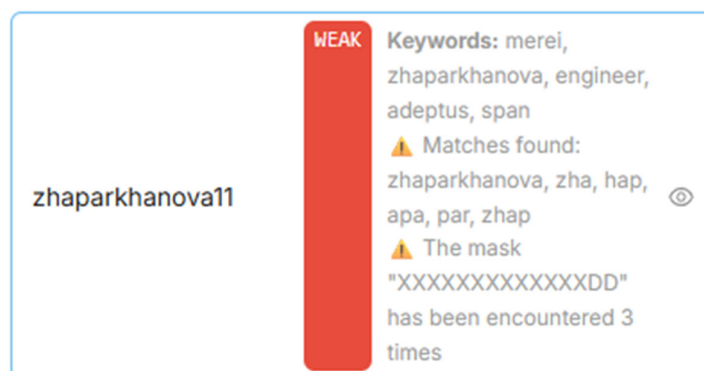


Figure 14. Chrome extension.

4. Evaluation and Results

To evaluate user-made password strength against brute force attacks, the LSTM model was trained on a personalized dataset. During training, it identified frequently repeated character sequences specific to an individual user.

Based on this, a list of the most frequent substrings was generated to help detect predictable and vulnerable passwords. Repeating fragments often suggest the use of names, birth years, or typical keyboard patterns. Below is an example of the detected pattern list:

```

"json"
{
  "merci": 20,
  "erici04": 1,
  "merci04": 1,
  "zha": 49,
  "apa": 96
}

```

Figure 15 shows a word cloud built from the most frequent substrings. The visualization highlights the prevalence of numeric fragments (e.g., ‘2004’) and personal tokens (e.g., “zhap”, “khan”, “Merei”), indicating a tendency to include names and dates in passwords.

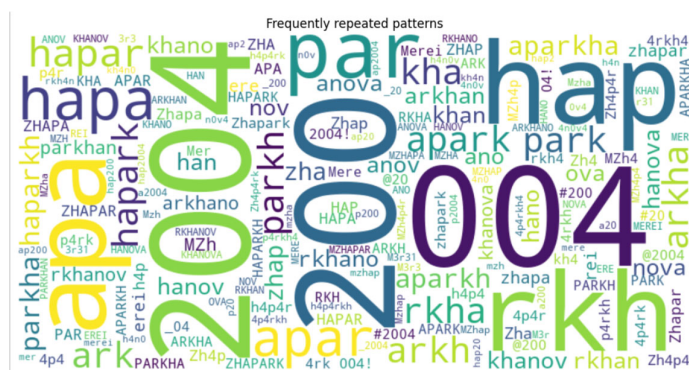


Figure 15. Frequently repeated patterns represented in word cloud.

This type of analysis is especially relevant when working with personalized data, as it exposes recurring structural weaknesses. These insights can later inform the development of personalized password-strength meters or adaptive security feedback systems.

Figure 8 demonstrates the convergence of the loss function during training. Both training and validation losses rapidly decreased in the first 10 epochs and stabilized below 0.15, indicating a stable and generalizable model. Figure 9 shows that the model reached a validation accuracy of 96.7%, with training accuracy closely matching, confirming the absence of overfitting.

In addition to classification accuracy, the average negative log-probability score (Avg Log-Score) was used to measure password uniqueness. The final average score on the validation set was 0.224, where lower scores indicate higher predictability. This score captures not only structural correctness but also semantic familiarity of patterns.

These results (Table 1) show that the LSTM model significantly outperforms classical models both in predictive accuracy and in capturing deeper structural dependencies of password composition. It is particularly effective in detecting behavioral patterns that are not explicitly rule-based.

Table 1. Model results comparison.

Model	Accuracy	Avg Log-Score	Adaptability	Manual Feature Design	Human Behavior Modeling
Decision Tree	81.4%	0.421	Low	Required	Absent
PCFG	88.2%	0.318	Medium	Required	Partial
LSTM	96.7%	0.224	High	Not required	Present

Finally, the connection between public personal information and password elements was examined through a method that utilized OSINT keyword extraction through the Serper API. The correlation heatmap (Figure 10) indicates that the relationship between passwords and personal names exhibited a robust positive correlation ($r \approx 0.65$), while partial string matches and semantic closeness also played a notable role in predictability. These results indicate that incorporating personal data in passwords continues to pose a significant security threat, even if users use slight alterations like inserting numbers or special characters.

Taken together, the evaluation outcomes validate that the LSTM-based model successfully grasps the structural and semantic characteristics of user passwords. It exceeds conventional methods in predictive capability and in its capacity to represent individualized patterns, thus providing a more smart and flexible solution for real-time password strength evaluation.

5. Discussion

Based on survey results [28], 84% of users use insecure passwords, relying on easily guessable and predictable sources of inspiration to create them.

The most common examples were the following: favorite number (24%), pet name (23%), date of birth of the user or their loved ones (19%), reuse of an old password (17%), and names of family members or partners (16%).

Also frequently encountered are the names of favorite movies, music groups, sports teams, memorable dates (8%), favorite color (8%), and even such trivial elements as the word “password” and its variations (4%).

These patterns (Figure 16) indicate that users are more often guided by their convenience and memorability than by information security requirements. As the survey

results show (see Figures 1–3), the majority of respondents admit that they try to minimize cognitive load by choosing simple, easy-to-remember combinations, often repeating or slightly modifying old passwords. The obligation to comply with formal requirements is perceived more as an onerous necessity than as a real protection measure.

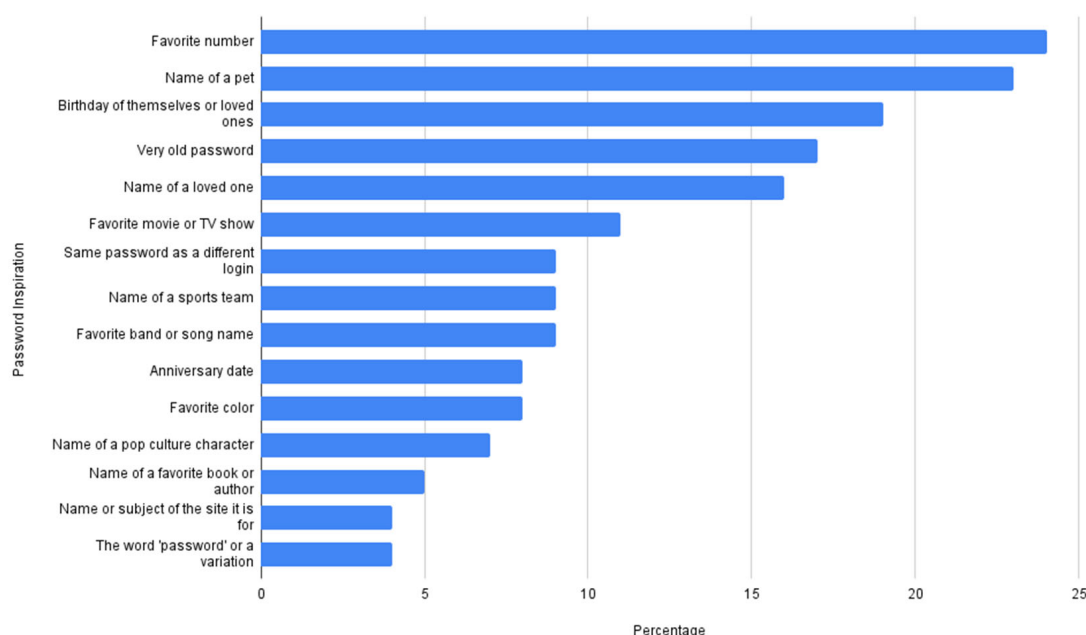


Figure 16. Frequency of occurrence of personal data in passwords.

The trend is especially pronounced in corporate environments where employees are pressed for time and often perceive cybersecurity requirements as a barrier to effective work performance. Under conditions of stress and high workload, even formally secure policies begin to be perceived as interfering, leading to superficial compliance: employees create passwords that formally comply with the rules (e.g., capitalization of letters and symbols) but are easy to guess (e.g., “Password2024!”).

Moreover, this behavior leads to the illusion of security: the system meets external criteria but remains vulnerable to attack. Rigid and inflexible requirements can lead to user resistance or encourage users to adopt workaround strategies that do not provide real security gains.

Existing online password evaluation services vary significantly in terms of the depth of their analysis and their approaches to verification. The most advanced of these is PasswordMonster [27], which simulates real attacks, including dictionary and substitution attacks (e.g., $a \rightarrow @$, $e \rightarrow 3$), analyzes sequences (12345, qwerty) and estimates the time required to crack the password. It also provides recommendations on digital hygiene, making it not only a diagnostic tool but also an educational one. In contrast, Kaspersky Password Checker [29] and Security.org [30] rely on simple heuristics (password length, presence of different types of characters) and do not take into account structural features or behavioral patterns. Another solution, UIC Password Checker [31], is a rule-based system limited by fixed criteria

Such methods are prone to errors and do not reflect the actual strength of a password. This highlights the need for intelligent adaptive approaches, such as neural network models, capable of identifying hidden patterns in the behavior of a specific user.

In addition, previously used methods, such as decision trees and probabilistic context-free grammars (PCFG), proved to be less flexible and limited in accounting for the peculiarities of human behavior when choosing passwords, compared to the LSTM model.

Decision trees are strictly dependent on predefined features and use clear logical divisions, which limits their ability to identify complex or hidden patterns. The PCFG model is more adaptive and allows for the probabilistic structure of passwords to be taken into account, but its effectiveness decreases with non-standard inputs and the absence of explicit grammatical rules (Table 2).

Table 2. Comparison of different approaches.

Criteria	Decision Tree	PCFG	PassGAN	LSTM
Model Type	Discrete, tree-based logic	Statistical grammar-based model	Generative adversarial network	Recurrent neural network
Sequence Processing	Not supported	Limited by predefined grammar rules	Fully supported, via generative sampling	Fully supported, context modeling
Requires Manual Feature Design	Yes	Yes	No	No
Flexibility with Novel Patterns	Low	Medium	High	High
Human Factor Consideration	Mostly not considered	Partially considered through probabilities	Indirect, inferred from data distribution	Present, models real user behavior
Robustness to Unconventional Data	Low	Medium	High	High
Interpretability	High	Medium	Very low (black-box generator)	Low (black-box model)
Password Representation	Logical rule tree	Structure + segment frequencies	Latent vector encoding	Embeddings and memory
Main Limitation	Requires manual feature selection	Cannot detect nested patterns	Lacks interpretability and user-specific feedback	Requires large data and training time

More advanced models, such as PassGAN [14], represent a further step forward, employing generative adversarial networks to synthesize human-like passwords based on learned distributions. However, while PassGAN [14] excels at generating realistic password candidates in bulk, it lacks the interpretability and structural analysis capabilities required for real-time feedback and user-specific vulnerability detection.

Unlike these approaches, the LSTM recurrent neural network is trained on sequential data and is capable of automatically identifying latent dependencies and contextual patterns without the need to manually specify feature structures or grammatical rules. This makes it more robust and effective when analyzing passwords formed under the influence of individual and behavioral factors.

The solution proposed in this paper, an LSTM-based password analysis model, answers this problem through a personalized approach. Instead of applying a single rule to everyone, the system analyzes the password structure, identifies recurring patterns and matches them with typical vulnerabilities, including the user's personal data. In this way, it provides feedback tailored to the individual's behavior, reducing cognitive load while increasing awareness and protection.

A promising avenue for future development involves the expanded application of profiling techniques through multimodal analysis of publicly available data from users' social media accounts, including LinkedIn, X (formerly Twitter), and Instagram. By lever-

aging additional AI agents, such as domain-specific natural language processors, vision models, and entity linkers, the system could be significantly enhanced in its ability to extract behavioral, contextual, and semantic features. These agents would allow for more granular and accurate modeling of a user's digital identity, enabling more precise password pattern inference and risk evaluation. The integration of such agents with the proposed framework could mark a shift towards more intelligent user modeling systems.

Another important direction for future research involves evaluating the long-term behavioral impact of our approach. A longitudinal study is planned to examine how the real-time feedback generated by the LSTM-driven password analysis tool influences users' future password creation strategies. This includes identifying which insecure behaviors—such as embedding personal data, using common patterns, or recycling old passwords with minimal modifications—users are most likely to abandon. Moreover, the study will assess the degree to which adaptive, context-sensitive feedback fosters the adoption of more secure, structurally diverse password habits. The findings may inform the design of adaptive behavioral cybersecurity tools that evolve in parallel with user tendencies, promoting long-term improvements in digital hygiene and self-aware security behavior.

6. Conclusions

This study introduces an LSTM-based password-strength model that fuses neural pattern detection with personalized behavioral insight. Deployed in both a web application and a Chrome extension, the system analyzes passwords locally, enriches its judgement with OSINT-derived personal tokens, and delivers real-time feedback that goes far beyond traditional length and character set rules. When benchmarked against classical methods (Section 4), the LSTM reached 96.7% validation accuracy and an average $-\log P$ of 0.224, outperforming a probabilistic context-free grammar (88.2%, 0.318) and a decision-tree baseline (81.4%, 0.421). These gains in performance, 15% greater than that of decision trees and 8.5% over PCFGs, while nearly halving predictability, confirm that learning sequential dependencies and user-specific cues yields a more reliable assessment and exposes hidden vulnerabilities that rule-based systems miss. This approach, therefore, boosts both accuracy and usability in personal, educational, and corporate settings, underlining the decisive role of human factors in security practices. Planned extensions, such as direct integration with password managers, adaptive user coaching, and optional multi-factor authentication checks, should broaden its reach and further promote digital hygiene while reducing real-world security risks. In conclusion, while the model provides meaningful insights into password patterns and user behavior, its client-side execution requires significant resources, which may limit accessibility on low-end devices. Additionally, the accuracy of its predictions is closely tied to the quality and recency of the training data. At this stage, automatic fine-tuning on the platform remains unimplemented, presenting an opportunity for future development.

Author Contributions: Conceptualization, L.R.; methodology, A.R., L.R. and M.Z.; software, M.Z. and Z.K.; validation, L.R.; formal analysis, Z.K.; investigation, Y.A.; resources, A.R.; data curation, A.I.; writing—original draft preparation, L.R., A.R. and M.Z.; writing—review and editing, L.R., A.M. and Y.A.; visualization, A.M., Y.A. and O.K. All authors have read and agreed to the published version of the manuscript.

Funding: This study was carried out with the financial support of the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan under Contract №388/PTF-24-26 dated 01.10.2024 under the scientific project IRN BR24993232 “Development of innovative technologies for conducting digital forensic investigations using intelligent software-hardware complexes”.

Institutional Review Board Statement: Ethical review and approval were waived for this study due to the anonymous nature of the survey and the voluntary participation of respondents.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The datasets used for training the model in this study are publicly available and properly cited in the References section. The survey data generated and analyzed during the current study are available within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
API	Application Programming Interface
Avg	Average
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
ML	Machine Learning
NIST	National Institute of Standards and Technology
ONNX	Open Neural Network Exchange
OWASP	Open Web Application Security Project
PCFG	Probabilistic Context-Free Grammar
RNN	Recurrent Neural Network
SMS	Short Message Service
SSPR	Self-Service Password Reset
TOTP	Time-Based One-Time Password
UCAPP	User-Centric Adaptive Password Policy
UI	User Interface
URL	Uniform Resource Locator
WASM	WebAssembly

References

1. Bergeron, A.; Dearden, T.E. Secret sharing in online communities: A comparative analysis of offender and non-offender password creation strategies. *J. Econ. Criminol.* **2024**, *6*, 100110. [CrossRef]
2. Atzori, M.; Calò, E.; Caruccio, L.; Cirillo, S.; Polese, G.; Solimando, G. Evaluating password strength based on information spread on social networks: A combined approach relying on data reconstruction and generative models. *Online Soc. Netw. Media* **2024**, *42*, 100278. [CrossRef]
3. He, D.; Yu, H.; Zhou, B.; Zhu, S.; Zhang, M.; Chan, S.; Guizani, M. How does social behavior affect your password? *IEEE Netw.* **2021**, *35*, 284–289. [CrossRef]
4. Buckman, B. 36 Must-Know Password Statistics for 2025 | Huntress. Available online: <https://www.huntress.com/blog/password-statistics/> (accessed on 27 July 2025).
5. Data Breach Statistics in 2024—Surfshark. (9 December 2020). Surfshark. Available online: <https://surfshark.com/research/study/data-breach-recap-2024> (accessed on 27 July 2025).
6. Urrico, R. Reports Show Rising Ransomware Attacks and Bad Password Habits Threaten Financial Accounts, Among Others. Finopotamus. 2024. Available online: <https://www.finopotamus.com/post/reports-show-rising-ransomware-attacks-and-bad-password-habits-threaten-financial-accounts-among-ot> (accessed on 27 July 2025).

7. Florêncio, D.; Herley, C. A large-scale study of web password habits. In Proceedings of the 16th International Conference on World Wide Web; Association for Computing Machinery: New York, NY, USA, 2007; pp. 657–666. [CrossRef]
8. Kävrestad, J.; Eriksson, F.; Nohlberg, M. Understanding passwords: A taxonomy of password creation strategies. *Inf. Comput. Secur.* **2019**, *27*, 453–467. [CrossRef]
9. Bonneau, J.; Herley, C.; van Oorschot, P.C.; Stajano, F. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 553–567. [CrossRef]
10. Komanduri, S.; Shay, R.; Kelley, P.G.; Mazurek, M.L.; Bauer, L.; Christin, N.; Cranor, L.F.; Egelman, S. Of passwords and people: Measuring the effect of password-composition policies. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada, 7–12 May 2011; pp. 2595–2604. [CrossRef]
11. National Institute of Standards and Technology. *Digital Identity Guidelines: Authentication and Lifecycle Management (NIST SP 800-63B, Rev. 4)*; U.S. Department of Commerce: Washington, DC, USA, 2024. [CrossRef]
12. Houshmand, S.; Aggarwal, S. Building better passwords using probabilistic techniques. In Proceedings of the 28th Annual Computer Security Applications Conference, Orlando, FL, USA, 3–7 December 2012; pp. 241–250. [CrossRef]
13. Melicher, W.; Ur, B.; Segreti, S.M.; Komanduri, S.; Shay, R.; Bauer, L.; Christin, N.; Cranor, L.F. Fast, lean, and accurate: Modeling password guessability using neural networks. In Proceedings of the 25th USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016; pp. 175–191. Available online: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher> (accessed on 27 July 2025).
14. Hitaj, B.; Gasti, P.; Ateniese, G.; Perez-Cabo, A. PassGAN: A deep learning approach for password guessing. In *Advances in Information Security*; Biggio, A., Roli, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 79, pp. 1–20.
15. Wang, D.; Wang, P.; Wang, J.; Liu, J. Targeted online password guessing: An underestimated threat. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1242–1254. [CrossRef]
16. Golla, M.; Dürmuth, M. On the accuracy of password strength meters. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1567–1582. [CrossRef]
17. Xu, M.; Feng, Y.; Ji, S.; Deng, R.H. On the account security risks posed by password strength meters. *arXiv* **2025**. [CrossRef]
18. Tippe, P.; Berner, M. Evaluating Argon2 adoption and effectiveness in real-world software [White paper]. *arXiv* **2025**, arXiv:2504.17121. [CrossRef]
19. Dropbox. zxcvbn: Low-Budget Password Strength Estimation [Software]. GitHub. 2012. Available online: <https://github.com/dropbox/zxcvbn> (accessed on 27 July 2025).
20. Pearman, S.; Thomas, J.; Emami-Naeini, P.; Habib, H.; Bauer, L.; Christin, N.; Cranor, L.F.; Egelman, S.; Forget, A. Let’s Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS ’17), Dallas, TX, USA, 30 October–3 November 2017; pp. 295–310. Available online: <https://dl.acm.org/doi/10.1145/3133956.3133973> (accessed on 27 July 2025).
21. Markert, C.; Golla, M.; Dürmuth, M.; Holz, T. On the security of modern smartphone unlock PINs. In Proceedings of the 2020 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 18–21 May 2020; pp. 1613–1630. [CrossRef]
22. Morag, Y.; Gutman, P.; Shir, E.; Kirchner, H. DPAR: Data-driven password advice and recommendations. In Proceedings of the 2022 IEEE European Symposium on Security and Privacy, Genoa, Italy, 6–10 June 2022; pp. 1–17. [CrossRef]
23. Glavin, L. Case Study: Microsoft. FIDO Alliance. 2025. Available online: <https://fidoalliance.org/case-study-microsoft/> (accessed on 27 July 2025).
24. Microsoft. Microsoft Digital Defense Report 2023. 2023. Available online: <https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2023> (accessed on 27 July 2025).
25. FIDO Alliance. The State of Passwordless: Progress Report. 2022. Available online: <https://fidoalliance.org/resource-library/> (accessed on 27 July 2025).
26. Password Dataset. Kaggle. 2024. Available online: <https://www.kaggle.com/datasets/soylevbeytullah/password-datas> (accessed on 27 July 2025).
27. Kkrypt0nn. Wordlists/Wordlists/Passwords at Main-Kkrypt0nn/Wordlists. GitHub. Available online: <https://github.com/kkrypt0nn/wordlists/tree/main/wordlists/passwords> (accessed on 27 July 2025).
28. All About Cookies. 84% of People Use Unsafe Passwords: Password Behavior Survey. 2023. Available online: <https://allaboutcookies.org/password-users-behavior-survey> (accessed on 27 July 2025).
29. PasswordMonster. Password Strength Meter. Available online: <https://www.passwordmonster.com/> (accessed on 27 July 2025).

30. Kaspersky Lab. Password Checker & Secure Random Password Generator. 2025. Available online: <https://password.kaspersky.com/> (accessed on 27 July 2025).
31. Security.org. How Secure Is My Password? | Password Strength Checker. 2025. Available online: <https://www.security.org/how-secure-is-my-password/> (accessed on 27 July 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.