Enhancing Password Security: A Quest for Optimal Honeywords

by

Meher Viswanath Nety

A Capstone Report submitted in
fulfillment of the requirements for the
degree of

Masters

in

Information Technology and Security- Artificial Intelligence

Ontario Tech University

Supervisor: Dr. Miguel Vargas Martin

October 2023

CAPSTONE RESEARCH PROJECT REVIEW INFORMATION

Submitted by: **Meher Viswanath Nety**

Master of Information Technology Security in Artificial Intelligence in Security field

Project/Major Paper title:

Enhancing Password Security: A Quest for Optimal Honeywords

The Project was approved on **17 November 2023** by the following review committee:

Review Committee:

Research Supervisor: **Dr. Miguel Vargas Martin**

Second Reader: **Dr. Patrick Hung**

The above review committee determined that the Project is acceptable in form and content and that a satisfactory knowledge of the field was covered by the work submitted. A copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

In this capstone report, our primary focus is on harnessing the capabilities of the GPT4 model to enhance password security through the generation of honeywords. Honeywords are decoy passwords designed to strengthen the security of sensitive systems by confusing potential attackers. The utilization of GPT4, a powerful language model developed by OpenAI, offers a n i nnovative a pproach t o t his c hallenge. By directly generating honeywords without relying on password segmentation, GPT4 introduces a unique dimension to password security. This approach is particularly valuable in thwarting targeted attacks, as honeywords generated by GPT4 are designed to deceive potential attackers effectively.

In addition to the exploration of GPT4, this report also delves into the realm of Chunk-GPT3. Chunk-GPT3, as detailed in previous research, employs advanced language models to generate honeywords through the segmentation of passwords into discrete chunks. These chunks are ingeniously recombined to form decoy passwords. The re-engineered Chunk-GPT3 approach incorporates enhancements to the password segmentation process, including "mapping digits to alphabets" and "removal of digits" functions. These modifications aim to produce more potent and effective honeywords, ultimately elevating password security.

The report includes a comprehensive comparative analysis of honeywords generated by the original Chunk-GPT3 approach and the re-engineered Chunk-GPT3

approach, as well as honeywords created by GPT4. By assessing the effectiveness of these honeyword generation methods using the HWSimilarity metric, the report provides valuable insights into the strengths and weaknesses of each approach. Examining the capabilities of both GPT4 and Chunk-GPT3 in the context of honeyword generation, this report aims to provide a holistic perspective on cutting-edge strategies for safeguarding sensitive data in the ever-evolving digital landscape.

Keywords: Authentication, Chunks, Honeywords, Passwords, Segmentation, GPT4

Author's Declaration

I hereby declare that this capstone project consists of original work authored by me. This is a true copy of the work, including all required final r evisions, a s accepted by my committee. I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this work to other institutions or individuals for scholarly research purposes. Additionally, I grant permission to the University of Ontario Institute of Technology (Ontario Tech University) to reproduce this work, either in whole or in part, by photocopying or other means, upon request from other institutions or individuals for scholarly research. I understand that my work may be made electronically available to the public.

——————————————————————————— Meher Viswanath Nety

Acknowledgements

Foremost, I express my deep gratitude to Dr. Miguel Vargas Martin, whose unwavering support and mentorship have been instrumental in shaping the direction of this endeavor. I greatly appreciate his belief in the significance of this research and his willingness to provide the necessary resources. I wish to extend my sincere appreciation to several individuals for their invaluable assistance in the successful completion of this project. Their contributions have been essential to its accomplishment.

I also want to recognize Fangyi Yu for her substantial role as the primary author of the Chunk-GPT3 paper. Her groundbreaking research lays the groundwork for this project, and I'm thankful for her innovative insights.

Satya Sannihith Lingutla deserves my deepest thanks for his unwavering support and invaluable contributions. His guidance and input have significantly enhanced the quality and depth of this work.

Furthermore, I want to express my heartfelt appreciation to all the individuals, colleagues, friends, and family members who have been by my side throughout this journey. Your enduring belief in my abilities has been a constant source of inspiration, and I'm genuinely grateful for your steadfast support.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the ever-evolving landscape of digital security, the imperative for robust password protection and efficient breach detection systems has grown exponentially. Traditional password databases have proven susceptible to breaches, leading to unauthorized access and potential exploitation of sensitive information. The concept of honeywords, or decoy passwords, introduced as a proactive measure alongside real passwords, has been proposed to expedite the detection of breaches and to minimize the time between compromise and response. The core idea underlying honeywords is that their use triggers an alert, indicating a potential data breach and prompting immediate countermeasures. However, the efficacy of honeywords hinges on their ability to closely mimic real-world passwords, making it a formidable challenge for attackers to distinguish between the two.

Passwords serve as the linchpin of network security, acting as the gatekeepers to authenticate users and grant access to valuable digital resources. Yet, passwords are not without their vulnerabilities and remain susceptible to a range of attacks, including dictionary attacks and brute-force attempts [7]. For the security of systems and the protection of sensitive data, the criteria for passwords are exacting. They

must be strong, unique, and confidential while remaining memorable for users. An ideal strong password typically encompasses a combination of lowercase and uppercase letters, special symbols, and digits, spanning a length of 8 to 12 characters [16].

In response to the limitations of traditional passwords, honeywords have emerged as a promising technique to enhance security. Honeywords are decoy passwords designed to closely resemble genuine passwords, with the sole purpose of triggering alerts when attackers attempt to use them during a breach. A plethora of Honeyword Generation Techniques (HGTs) have been proposed, each striving to create honeywords that maintain their resemblance to real passwords, even in the presence of personally identifiable information (PII) known to potential attackers [7] [16] [18].

1.1 Contribution

1. **Research Focus:** Our study explores Honeyword generation techniques, with a particular emphasis on Chunk-GPT3, and GPT4, an innovative approach employing advanced language models to enhance password security.

2. **Refinement of Chunking Algorithm:** In our research, we explored the impact of refining the chunking algorithm, specifically by employing the techniques of "mapping digits to letters" and "removal of digits." The aim was to assess whether these refinements could enhance the quality and effectiveness of Honeywords generated by Chunk-GPT3. Notably, our findings align with those of my previous capstone research, indicating that there was no substantial difference in the results.

3. **GPT4 Exploration:** We explore an alternative approach by investigating whether GPT4 can directly generate Honeywords without relying on the chunk-

ing algorithm, thus expanding the scope of our research.

4. **Comparative Analysis:** We aim to conduct a comprehensive comparative analysis, pitting Honeywords produced with the GPT4 which do not use a chunking algorithm to generate honeywords, and also optimized chunking algorithm against those generated by the original Chunk-GPT3 method to assess the impact of the enhanced chunking algorithm.

5. **Relevance to Password Security:** Our research aligns with the mission of strengthening password security and enhancing detection mechanisms for potential data breaches within online systems.

1.2 Capstone Report Structure

The upcoming sections of this document are structured as follows:

- In Section 2, we explore previous related works and explain our approach by outlining the changes made to the algorithm.
- Section 3 outlines our approach for generating honeywords using GPT4 without the need for chunks.
- Section 4 delves into the experiments conducted to produce honeywords using various models.
- Section 5 summarizes the results and evaluates HWSimilarity scores for Chunk-GPT3, Modified Chunk-GPT3, and GPT4.
- Section 6 presents suggestions for potential research directions in the future.
- Finally, in Section 7, we draw conclusions based on our research findings.

Chapter 2

Related Works

Numerous studies have explored the application of Natural Language Processing (NLP) methods to the segmentation and classification of password samples, seeking to enhance password security and understand the underlying patterns in users' password choices. In this context, one prominent approach is the utilization of NLP techniques to decompose passwords into coherent and meaningful parts, thereby enabling the inference of their syntactic function and semantic significance [15].

To address the syntactic classification required for this segmentation, part-of-speech tagging has been identified as an essential step in the process. This tagging procedure enables the algorithm to semantically classify nouns and verbs, refining the accuracy of the results. Notably, the outcomes have demonstrated a successful disambiguation of words based on contextual cues, as evident in passwords like "gangsterlove" and "ilovestacy," where the word "love" assumes distinct syntactic functions as a noun and a verb, respectively [14].

The researchers propose a new approach to understanding passwords, treating them as combinations of frequently occurring character sequences called "chunks." They create a specialized method to automatically segment passwords into these

chunks and then build three different models for guessing passwords at the chunk level using Markov, Probabilistic Context-free Grammar (PCFG), and neural networks [14] [4]. After testing their models on a massive dataset of 250 million passwords, the researchers observe substantial improvements in guessing efficiency. The Markov, PCFG, and neural network models enhance guessing efficiency by 5.7%, 51.2%, and 41.9% respectively in offline guessing scenarios. This underscores the importance of a suitable password representation in defending against attacks. The analysis of these efficient attacks reveals that common chunks within passwords serve as stronger indicators of vulnerability compared to the complexity of character classes.

Many studies have investigated honeyword generation techniques (HGTs) for enhancing password security. HGTs can be categorized into chaffing-by-tweaking and chaffing-with-a-password-model. Chaffing-by-tweaking involves substituting random letters, digits, and symbols in real passwords to create honeywords. However, this method has been proven vulnerable to attacks. On the other hand, the chaffing-with-a-password-model, exemplified by Kamouflage, tokenizes real passwords and substitutes each token with a random one that matches its type. While this approach is more resistant to attacks, it requires substantial modifications to the authentication system, impacting usability, and is limited in generating honeywords of varying lengths and structures.

From one of the HGTs, we focus on the potential security threat presented by GPT3 in its ability to distinguish actual passwords from honeywords and the strategies proposed to address this challenge. The authors of the paper highlight three key attributes of GPT3, emphasizing the importance of a clear prompt, temperature settings, and zero-shot and few-shot learning. The process of distinguishing actual passwords from honeywords involves setting prompts and temperatures, creating sweet word lists, and using different prompts to predict the actual password from

the list. These steps were evaluated using real-world password datasets, myspace1, and webhost2, to assess GPT3's performance. It's important to note that GPT3 is a paid service and its cost increases with the number of passwords involved in the analysis. Due to budget constraints, the analysis in the paper was limited to a subset of the datasets. The paper's findings are significant, as they highlight the security concerns associated with GPT3. Attackers can reduce the "trap size" by nearly 50 percent in some scenarios, posing a substantial threat to security. GPT3 also demonstrated a high success rate in capturing actual passwords, underscoring the need for proactive measures. To mitigate this security risk, the paper suggests that system administrators have the flexibility to adjust system-wide security thresholds based on their observations of the attacker's efficacy and dataset size. This adjustment can reduce the attacker's success probability by a factor , enhancing overall security. [3]

For targeted honeyword generation, the challenge is to split real passwords into tokens, retaining personally identifiable information (PII) tokens while replacing non-PII tokens with random ones. To address this, a chunking algorithm is proposed to divide passwords into semantic chunks, and a pre-trained generative model is used to create honeywords based on these semantic chunks [4].

Overall, the research in this area highlights the importance of developing stronger password chunks and develop better honeywords. However, challenges remain in balancing security and usability while effectively creating honeywords that closely resemble real passwords.

2.1 DataSet

The dataset comprises 1.4 billion email-password pairs sourced from various data breaches discovered on the DarkWeb2 in December 2017. These breaches include

data from well-known websites like Canva, Chegg, Dropbox, LinkedIn, and Yahoo! To ensure uniqueness, duplicate email-password pairs were removed, resulting in 1.1 billion distinct emails and 463 million distinct passwords. For simplicity, only the prefixes of email addresses were considered as usernames [15].

To obtain genuine passwords, those that were too short (less than 8 characters) or too long (more than 32 characters) were excluded, as they are typically not allowed by most authentication systems. This screening process yielded a final selection of 28,492 username-password pairs. The strength of each password was assessed using the "zxcvbn" tool, revealing that 24,661 passwords had a zxcvbn score of 4, 2706 passwords scored 3, and 277 and 3 passwords had scores of 1 and 0, respectively.

Further, two sets of username-password combinations were created based on the zxcvbn password scores. The first set called the zxcvbn-weak set, consisted of 1000 username-password pairs with the lowest zxcvbn scores, ranging from 0 to 2. The second set, known as the zxcvbn-strong set, comprised 1000 username-password pairs with the highest zxcvbn score of 4. Subsequently, the chunks in these two sets were further analyzed and compared, and honeywords were generated using the proposed method [16].

2.2 Honeyword-Generation Techniques

This section delves into the intricate world of honeyword-generation algorithms, a crucial component of enhancing password security. These algorithms fall into two primary categories: those that are independent of specific passwords and those that depend on the characteristics of account passwords.

2.2.1 Password-Independent Honeyword Generation

Password-independent algorithms are designed to create honeywords without relying on individual account passwords. Instead, they draw inspiration from password models that have been pre-trained on a wide array of passwords. Four prominent password models play a pivotal role in this approach: the list model, probabilistic context-free grammar model, Markov model, and recurrent neural network, alongside various combinations thereof. These models lay the groundwork for generating honeywords that effectively confound potential attackers. In the context of this approach, the algorithms are denoted as below, PCFG, Markov, RNN, and Combo.

2.2.2 Password-Dependent Honeyword Generation

In contrast, password-dependent algorithms take into account the difference of specific account passwords when generating honeywords. These algorithms encompass two distinct subcategories: password-strength-dependent and password-context-dependent methods. Password-strength-dependent methods strive to ensure that generated honeywords match the strength of the input password. However, recognizing the challenges posed by weaker passwords, these methods relax the requirement of exact strength matching, shifting the focus to matching the length of the input password.

The latter category, password-context-dependent methods, introduces modifications to the input passwords to generate honeywords. This section explores four distinct techniques within this category:

- Targeted password model-based generation
- Large language model-based generation (LLM)
- Random replacement-based tweaking

- Deep neural network (DNN)-based tweaking.

Targeted password model-based generation leverages password templates to craft honeywords. LLM-based generation involves querying extensive language models, such as GPT3, using prompts derived from the input passwords. Random replacement-based tweaking entails making random alterations to the characters of the input password, while DNN-based tweaking harnesses deep neural networks to create honeywords by tweaking the original password. Each technique within these methods is distinctly labeled by appending specific symbols to the relevant password model. [6]

2.3 Chunk-GPT3

Chunk-GPT3 serves as a technique for generating honeywords, using the GPT model. One of the primary challenges in constructing a Honeyword Generation Technique (HGT) is to produce honeywords that can withstand targeted attacks. Such targeted attacks are a significant concern in the realm of cybersecurity, as malicious actors often exploit individuals' personally identifiable information (PII) to guess their passwords, elevating the risk of unauthorized account access. This problem is exacerbated by the prevalence of data breaches, which make vast amounts of PII and passwords readily available to attackers. If the honeywords generated by the HGT do not take PII into account, the attacker's ability to identify the password becomes considerably higher. Chunk-GPT3, as an HGT, specializes in generating honeywords that resist targeted attacks.

The initial step involves assessing password strength, which is accomplished using the zxcvbn library. Subsequently, the passwords are categorized based on their zxcvbn strength and divided into two datasets, one containing strong passwords and the other weak passwords. These datasets then undergo a process using the Pwd

segment chunking algorithm to create password chunks for honeyword generation. The chief advantage of this chunking algorithm lies in its ability to generate chunks that exclusively consist of original password segments. To ensure that honeywords are resilient to targeted attacks, GPT3 is employed to generate them by providing prompts to the model.

Three crucial factors influence the generation of effective honeywords: the prompt, the temperature, and the examples provided to the model (the generated chunks). The temperature, a numeric value that can be set to 0 or 1, regulates the model's confidence level in producing accurate predictions. For optimal results, it is highly recommended to keep the temperature at 1. Therefore, the password is divided into chunks and presented as input to the model, such as requesting, "Please generate three passwords similar to 'Elena1986@327' and containing 'Elena,' '1986,' and '327'.

The evaluation of this approach is performed using flatness and success-number-graphs. In a comparison with other HGTs, namely chaffing-by-tweaking and chaffing-by-fasttext, Chunk-GPT3 excels by achieving a remarkable 85 percent similarity score with honeywords. A notable limitation of this model is the significance of accurately segmenting passwords. Incorrect segmentation can potentially provide attackers with easier access to the password. [18]

2.4 Re-engineering Password Chunking Algorithm

Improving Chunk-GPT3 Model

In the Chunk-GPT3 model, we create chunks from passwords using a technique called "PwdSegment." PwdSegment is an algorithm that employs byte-pair encoding (BPE) to break down plaintext passwords into chunk vocabularies. BPE is a commonly used method in natural language processing (NLP) to split words into smaller parts.

PwdSegment enhances BPE by allowing us to control the level of segmentation using an average length parameter. It counts character pairs and stops combining them when the resulting chunk vocabulary reaches the specified threshold length. The algorithm keeps merging the most common character pairs until the average length of the chunk vocabulary matches the threshold. This approach provides flexibility in generating chunk vocabularies for password analysis. However, PwdSegment has a limitation – it may not capture pieces of personally identifiable information (PII) within passwords [17].

For example, consider the password "h2omegatania." PwdSegment would create the chunks 'h2o', 'mega', '-', and 'tania', but it might miss the possibility that omega or home could contain PII. This limitation directly affects Chunk-GPT3. To address this, we re-engineered the PwdSegment algorithm to generate a larger number of potential chunks, with the expectation that this would result in higher-quality honeywords.

In the first phase of Capstone, our research focused on re-engineering the Password Chunking Algorithm. This re-engineering focuses on enhancing the code used for analyzing password fragments. The modified system introduces new capabilities, including digit-to-alphabet mapping, digit deletion, and the analysis of plain text passwords. These changes expand the code's functionality beyond its initial focus on analyzing both strong and weak passwords.

2.4.1 Mapping Digits to Alphabets

We introduced a new feature called "Digit-to-Alphabet Mapping." This feature associates individual digits in passwords with their corresponding alphabetic characters.

For instance, in the first scenario, passwords like "h0mega-tania" with a single digit get mapped to its corresponding alphabetic character. In the second scenario,

Digit	Alphabet
0	o
1	l
2	z
3	e
4	a
5	s
6	g
7	t
8	b
9	g

Table 2.1: Mapping Digits to Alphabets.

passwords such as "h20mega-tania" with two consecutive digits are mapped accordingly. However, in the third scenario, passwords like "h123mega-tania" with three consecutive digits remain unchanged. This is because passwords with at least one instance of three or more consecutive digits, like "h123mega-tania," keep the numbers in their original form since they might contain personally identifiable information (PII). Similarly, if a password like "h123mega-98tania" contains three consecutive digits in one position and two consecutive digits in another, it remains unchanged.

Original Password	Mapped Password
h0mega-tania	homega-tania
h20mega-tania	hzomega-tania
h123mega-tania	h123mega-tania

Table 2.2: Password Mapping

2.4.2 Removal of Digits

In addition to mapping single digits to their corresponding characters, as explained above, we also modified passwords by removing single digits. This involves elimi-

nating a single numerical character, resulting in plain text passwords without any numeric elements. By removing the single digit, it becomes possible to analyze passwords independently, without considering the presence or significance of the numeric element. The resulting password, with the single digit removed, is referred to as a "Text Password." Using this method, the original password "h20mega-tania" is transformed into the corresponding text password "hmega-tania" (in addition to the "Digit-to-Alphabet Mapping" mentioned earlier). This enables the evaluation and examination of the password's security without considering the presence of a single numerical digit.

2.4.3 Password Chunk Analysis

Our re-engineered code conducts chunk analysis on three different types of passwords: original passwords, mapped passwords, and text passwords. Here are the steps taken for each type:

1. **Original Passwords:** We start by reading original passwords from two CSV files. One file contains 1000 zxcvbn-strong passwords, and the other contains 1000 zxcvbn-weak passwords. I add a new column to these CSV files, which contains password chunks extracted using the Password Strength Meter (PSM) library. This algorithm divides the passwords into smaller chunks, making it easier to assess their strength accurately. Another column is added to count the number of password chunks in each password. We then use these new columns to create two additional CSV files, one for zxcvbn-strong passwords and another for zxcvbn-weak passwords. These files contain the passwords and their respective chunks, along with the count of chunks per password.

2. **Mapped Passwords:** Our re-engineered method introduces the concept of

	username	pw	strength	pw_chunks	num_pwchunks	mpw	mpw_chunks	num_mpchunks	tpw	tpw_chunks	num_tpchunks	chunks	num_chunks
0	00-00-00-00	данияр123456789101112131415	4	{'данияр', '123456789101112131415'}	2	данияр123456789101112131415	{'данияр', '123456789101112131415'}	2	данияр	{'данияр'}	1	{'данияр', '123456789101112131415'}	2
1	y5537787t	emperorpalpateen	4	{'teen', 'pa', 'pal', 'emperor'}	4	emperorpalpateen	{'teen', 'pa', 'pal', 'emperor'}	4	emperorpalpateen	{'teen', 'pa', 'pal', 'emperor'}	4	{'teen', 'pa', 'pal', 'emperor'}	4
2	i887lo5412va612	4ерер89652301	4	{'ер', '896523', '4е', '01'}	4	4ерер89652301	{'ер', '896523', '4е', '01'}	4	ерер	{'ерер'}	1	{'ерер', '4е', '01', '896523', 'ер'}	5
3	c21marella	c21marella@gmail	4	{'@gmail', 'marella', 'c21'}	3	czimarella@gmail	{'@gmail', 'czi', 'marella'}	3	cmarella@gmail	{'c', '@gmail', 'marella'}	3	{'czi', '@gmail', 'c', 'marella', 'c21'}	5
4	a1069268	a1069268@bofthew.com	4	{'com', '@', 'a', 'bofthew', '.', '1069268'}	6	a1069268@bofthew.com	{'com', '@', 'a', 'bofthew', '.', '1069268'}	6	a@bofthew.com	{'bof', '.com', 'thew', 'a@'}	4	{'.com', 'com', '@', 'a', 'a@', 'bof', 'bofthe...	10

Figure 2.1: Data Frame of the columns related to original, mapped, and text password chunks of Strong passwords.

	username	pw	strength	pw_chunks	num_pwchunks	mpw	mpw_chunks	num_mpchunks	tpw	tpw_chunks	num_tpchunks	chunks	num_chunks
0	s1lihceyhuna	1s1s1s1s1s1s1s1s	0	{'s', '1', 's1'}	3	isisisisisisisis	{'isisisis'}	1	sssssssss	{'sssssssss'}	1	{'isisisis', 's', '1', 's1', 'sssssssss'}	5
1	w1e	chicagochicagochicago	0	{'chicago'}	1	chicagochicagochicago	{'chicago'}	1	chicagochicagochicago	{'chicago'}	1	{'chicago'}	1
2	q0q0q0q0q0q0q0q0	q0q0q0q0q0q0q0q0	0	{'q0'}	1	qoqoqoqoqoqoqoqo	{'qo', 'o', 'oq', 'qoq'}	4	qqqqqqqqq	{'qqqqqqqqq'}	1	{'oq', 'o', 'q0', 'qo', 'qqqqqqqq', 'qoq'}	6
3	w1w2w3	t64t64t64t64t64t64t64t64t64t64	1	{'64', 't'}	2	tgatgatgatgatgatgatgatgatga	{'atg', 'ga', 'at', 'tg'}	4	ttttttttt	{'ttttttttt'}	1	{'ttttttttt', '64', 't', 'ga', 'at', 'tg', 'a...	7
4	k1a2t3e4	katrina999666	1	{'999666', 'katrina'}	2	katrina999666	{'999666', 'katrina'}	2	katrina	{'katrina'}	1	{'999666', 'katrina'}	2

Figure 2.2: Data Frame of the columns related to original, mapped, and text password chunks of Weak passwords.

transformed passwords by applying the Digit-to-Alphabet mapping to the original passwords. The resulting mapped passwords are stored in a new column. Similar to the process for original passwords, we create two new CSV files with the chunks and the number of chunks per password.

3. **Text Passwords:** In our re-engineered approach, we process original passwords using the "Removal of Digits" method described earlier. Just like the process applied to original and mapped passwords, we generate the chunks and record the number of chunks per password.

2.4.4 Appending Chunks

The re-engineered method consolidates all chunks obtained for each of the original passwords, including mapped and plain text passwords. We have divided the data into two figures. These additions make it easy to compare and analyze password representations, facilitating a comprehensive evaluation of password strength.

2.4.5 Honeyword Generation using Re-engineered PwdSegment Chunks

We utilize the Chunk-GPT3 model to generate honeywords using the re-engineered chunks. These honeywords are generated using the same prompts as in a previous study [18], which include the original password, its chunks, and a length limit.

2.4.6 Results for Re-engineered PwdSegment Chunks

In the first phase of the capstone, we evaluated the performance of our improved chunking algorithm by using a metric called 'HWSimilarity.' This metric helps us measure the similarity between the honeywords generated with our enhanced method and the original passwords. Here are the key findings:

- **Strong Passwords:** The improved chunking algorithm achieved a HWSimilarity score of 0.8409, slightly lower than the original chunking algorithm's score of 0.8525. This indicates that the honeywords created by my method are somewhat more distinct from the original strong passwords.

- **Weak Passwords:** Notably, for weak passwords, the improved chunking algorithm outperformed the original one with a score of 0.9048, compared to the original algorithm's score of 0.8367. This signifies that the enhanced approach excels at generating honeywords that are significantly different from the original weak passwords, providing enhanced security for these accounts.

In summary, our improved chunking algorithm, while slightly less similar to strong passwords, is substantially more effective at generating distinct honeywords for weak passwords. This means our approach offers an additional layer of security for accounts with weaker passwords, which are often more susceptible to security threats.

Chapter 3

Approach

We embarked on a comprehensive exploration of various honeyword generation techniques to bolster password security. This journey began in the first phase of capstone project, where we focused on re-engineering the chunking algorithm. The modified algorithm, termed "Re-engineered Chunk-GPT3," harnessed advanced language models to create honeywords by segmenting passwords into distinct chunks, cleverly recombining them to form decoy passwords. Notably, this approach exhibited remarkable resilience against targeted attacks, showing promising potential to enhance password security significantly [18].

In second phase of capstone project, we shifted my research focus towards the evaluation of GPT4's capability to directly generate honeywords without relying on chunking. We examined whether honeywords generated by GPT4 were more potent than those created with Chunk-GPT3. This approach introduced a novel dimension to our research, offering insights into diverse methods for fortifying password security.

To gauge the strength of the honeywords generated, we utilized the HWSimilarity metric. This metric serves as a valuable tool to compare and assess honeywords produced using different techniques. Its primary function is to measure how closely

honeywords resemble real passwords, making it challenging for potential attackers to distinguish between them. The metric assigns a similarity score to each honeyword, reflecting the degree to which it mirrors actual password attributes. Higher scores indicate a stronger resemblance, while lower scores suggest less convincing honeywords.

My research involved conducting a comprehensive comparative analysis between honeywords generated by the original Chunk-GPT3, honeywords produced by the re-engineered Chunk-GPT3 approach, and those created directly by GPT4. This evaluation aimed to determine the effectiveness of GPT4 in generating more potent and effective honeywords. In doing so, our research aligned with the broader mission of enhancing password security and strengthening the detection mechanisms for potential data breaches in online systems.

As the digital landscape continues to evolve, safeguarding sensitive data remains a top priority. This thesis represents a significant step towards achieving heightened password security, offering fresh insights and innovative solutions crucial for countering ever-evolving cyber threats.

3.1 Honeywords Generated by GPT4

The code we created uses a smart tool called GPT4 from OpenAI to generate fake passwords, known as honeywords. These honeywords are like decoys, making it hard for hackers to figure out the real passwords. We tested this code on two types of passwords: strong ones, weak ones The main focus was comparing the honeywords made for strong and weak passwords and seeing how changing passwords affected the results.

Key Components of the Developed Code:

1. Honeyword Generation Function

    - This function utilizes OpenAI's GPT4 model to generate honeywords for a given real password.
    - It constructs a chat-based message with the user role, providing a prompt to derive 19 distinct passwords similar to the real password.
    - The generated honeywords adhere to specified criteria, including a maximum length equal to the original password's length and the absence of digits at the end.
    - The GPT4 response is processed, extracting and cleaning the generated honeywords based on a defined length threshold.
    - The function returns up to 19 cleaned honeywords.

2. Honeyword Generation for Password Sets ('chaffing by GPT4')

    - This function iterates through a DataFrame containing passwords (with optional chunks, which have been removed in this version of the code).
    - For each password in the DataFrame, the 'honeywords generation GPT4' function is called to generate honeywords.
    - The generated honeywords are stored in a dictionary with the real passwords as keys.

3. Category-Specific Honeyword Generation

    - The code generates honeywords for the first 100 passwords from the strong and weak password Data Frames as shown in the below figures. Adjustments can be made to accommodate a larger dataset.

0	1	2	3	4	5	6	7
данияр123456789101112131415	данияр246813579111315	данияр123456789102014	данияр123456789101212	данияр12345678111215	данияр1357911234567	данияр24681012141618	данияр12345678101923
emperorpalpateen	EmperorPlapateen	EmperorP@lpateen	Emper0rPalpateen	Emp3rorP@lpateen	Emper0rP@lpateen	EmperorP@lpat33n	EmperorPalpat33n
4erep89652301	4erp689652301	4erep89625130	4erep89632150	4er896ep52301	4eer896p52301	4e94rep896523	erep489652301
c21marella@gmail	c21marellaPlus	Unique21marella	Kingc21marella	Marellac21day	Springc21marrla	c21SweetyMarella	M21CarellaGmail
a1069268@bofthew.com	Bof1069268@thew.com	Bofa1069268@thw.com	Ab1069268@ofthew.com	1069268A@bofthew.com	A1069268B@ofthw.com	BofA1069268@thew.com	Bofa_1069268@thew.com
peregrebnoe2007	Peregr2007ebno	Pere07grebnoe	20Peregrebnoe7	pereGREB2007no	PerEgrEBnoe07	PeregrEB2007No	2007PereGrebno
5,131,424,096	guidelines,	passwords	5,131,424,096:	5,131,424-096	513142-4096	5.131.424.096	5131.424.096
314,945,480,683,060	934,415,408,683,620	314,495,408,683,960	681,945,380,723,014	314,598,470,680,043	450,319,480,693,614	341,945,083,680,610	433,194,854,086,310
vnbriuhj44839	Vnbriuhj448321	VNBr${#44839	vnbR!uhJ448^	vNbrUhjF4483	vnbrIuhJ83k9	448vNbriUhj39	vN#4Br!uHj83
"fbobh_yq3"""	fbob_yq3h	f_bobhyq3	f3bobyqh_	fb_obhyq3	fybob_qh3	fb3obh_yq	hfbob_y3q

Figure 3.1: 100 Generated Honeywords for strong passwords

0	1	2	3	4	5	6	7	8	9
1s1s1s1s1s1s1s1s	1a1a1a1a1a1a	1b1b1b1b1b1b	1c1c1c1c1c1c	1d1d1d1d1d1d	1e1e1e1e1e1e	1f1f1f1f1f1f	1g1g1g1g1g1g	1h1h1h1h1h1h	1i1i1i1i1i1i
chicagochicagochicago	chicagochicago1go	1chicagochicagochi	chicago2chicagochi	2chicagochicagogo	chicagochicago3hi	3chicagochicagoch	chicago4hicagochi	4chicagochicagogo	chic5agochic
q0q0q0q0q0q0q0q0	qoq0q0q0o0q0q0	0q0q0q0q0q0q0q	q00q00q00q0q0q	qoqoqoqoqoqoqo	0q0q0q0q0q0oq	pqpqpqpqpqpqpqp	qoq0qoq0qoq0qo	oqoqoqoqoqoqoq	qpqpqpqpqp
t64t64t64t64t64t64t64t64t64t64	4t64t64t64t64t64t64t6	t64t64t64t64t64t64t9	t64t64t64t64t64t9t64	t64t64t64t64t9t64t64	t64t64t64t9t64t64t64	t64t64t9t64t64t64t64	t64t9t64t64t64t64t64	t9t64t64t64t64t64t64	1t64t64t64t6
katrina999666	katrina966699	katrina669996	katrina969696	katrina666999	999katrina666	666katrina999	7999katrina66	6699katrina96	8999katrina6
123012301230GG	1230123012GG3	1201230123OGG	11230123012GG	G12301230123G	GG12301230123	123G12301230G	1230G1230123G	12301G230123G	1230123G12
timberwolvesD	TimberwolvesZ	wolveTimberD	WolvesTimberQ	Dtimberwolves	voxTimberWolf	TimberwolveKs	wolvesTimberU	TimberwolvesA	wolvesDTimb
123456h7777777	123456h777777	12345h6777777	12344h6777777	123456h8777777	123456h7777776	12233456h7777777	12345566h7777777	1234456h7777777	12356h7777
ELLAELLA159159	ELLAUmbrella159	Ella159Ella	E9llaeLLa159	Ella15a159	Ellella15N159	ElLa151ELLA59	Ellaella1Space59	EllaEllaZ159159	E11AElla158
lascolascolasco	lascolascolasc	lascolascolas	laslascolasco	Coslascolasco	lascolaslasco	lasclascolasc	lasColascOlas	lascolascOias	Ascolascola

Figure 3.2: 100 Generated Honeywords for weak passwords

- The generated honeywords, along with their corresponding real passwords, are organized into a structured response.

Chapter 4

Experiments

4.1 Honeywords Generation

Honeyword generation using language models like GPT3 and GPT4 represents an innovative approach to enhancing password security. These models leverage the natural language understanding capabilities and text generation capabilities to create convincing decoy passwords, known as honeywords. Honeywords are designed to closely mimic real passwords and are interspersed within a password database to increase security by confusing potential attackers.

4.1.1 Importance of Password Security

Passwords are a fundamental component of digital security, and their compromise can lead to significant data breaches and unauthorized access. The need for robust password security is underscored by the prevalence of cyberattacks. [1]

4.1.2 Honeywords as a Security Measure

Honeywords add an extra layer of security to password databases. By introducing convincing decoy passwords alongside real ones, honeywords make it challenging for attackers to discern genuine passwords, thus improving the security of systems.

4.1.3 Role of AI and Language Models

AI-driven language models like GPT3 and GPT4 have shown remarkable capabilities in natural language understanding and text generation. They are being harnessed to generate honeywords that closely resemble real passwords, making it difficult for attackers to distinguish between the two.

4.1.4 Customization and Fine-Tuning

Researchers and organizations often fine-tune language models to generate honeywords tailored to specific security requirements. This customization allows for the creation of honeywords that align with the unique characteristics of a system or password database. [8]

4.1.5 Evaluating Honeyword Quality

The effectiveness of honeywords is typically evaluated using metrics like HWSimilarity, which assess how closely honeywords resemble real passwords. A higher HWSimilarity score indicates a better honeyword generation technique. [9]

4.1.6 Strengths and Limitations

While GPT3 and GPT4 offer powerful tools for honeyword generation, there may be limitations such as scalability and fine-tuning requirements. Researchers continue

to explore the strengths and weaknesses of these models in the context of password security. [10]

Incorporating GPT3 and GPT4 in honeyword generation is a promising step in bolstering password security, and researchers are actively working to harness the full potential of these models in safeguarding digital systems and data.

4.2 Honeywords Generation using Chunk-GPT3

4.2.1 Enhancing Password Security with Honeywords Generated by Chunk-GPT3

In today's digital age, safeguarding sensitive information is of paramount importance. Passwords are a fundamental aspect of this security framework, serving as a primary barrier against unauthorized access to personal, corporate, or governmental systems. However, the traditional approach to password security, which relies solely on the secrecy of passwords, is increasingly vulnerable to various forms of attacks. Password breaches, phishing attacks, and the prevalence of weak and easily guessable passwords highlight the critical need for advanced security measures.

4.2.2 Honeywords and their Significance

One innovative solution that has emerged to bolster password security is the concept of "honeywords." Honeywords are essentially decoy passwords designed to confuse potential attackers. They are inserted into password databases alongside real passwords and are not meant to be used by legitimate users. The aim is to create a situation where an attacker who gains access to the database is unable to distinguish real passwords from honeywords, thereby increasing the uncertainty of the attack. Hon-

eywords are a proactive defense mechanism, designed not only to detect unauthorized access but also to deter and disrupt attackers in their tracks.

4.2.3 Chunk-GPT3 as a Tool for Honeyword Generation

Generating honeywords that effectively mimic real passwords, and yet remain distinct and secure, is a challenging task. This is where artificial intelligence, particularly OpenAI's GPT3 model, comes into play. GPT3 is a powerful language model that has been trained on vast amounts of text data, enabling it to generate human-like text and understand context.

The code presented in this thesis utilizes OpenAI's GPT3 to generate honeywords for real passwords. By providing GPT3 with a real password and associated password chunks, the code prompts the model to produce a set of honeywords that resemble the real password while adhering to specified criteria, such as length and character constraints. GPT3's natural language generation capabilities make it a valuable tool for this task, as it can generate honeywords that are contextually convincing and, thus, more likely to confuse potential attackers.

4.2.4 The Significance of This Approach

The utilization of GPT3 for honeyword generation offers several benefits. It allows for the automatic and rapid creation of a large number of honeywords, making it a scalable solution for enhancing password security. Furthermore, GPT3's ability to generate contextually relevant honeywords adds a layer of complexity that can significantly deter attackers. By providing a robust defense against password breaches, phishing attempts, and other forms of cyberattacks, honeywords generated by GPT3 contribute to overall information security and data protection.

The code is designed to generate honeywords, which are fake or decoy passwords used to enhance security by confusing potential attackers. It employs OpenAI's GPT3 model to create these honeywords.

1. Honeyword Generation Process: The code consists of two main functions: 'GPT3 honeywords' and 'chaffing by GPT3'.

- GPT3 honeywords: This function generates honeywords for a given real password and a set of password chunks. It uses GPT3 to generate similar passwords that adhere to certain criteria, such as length and the absence of digits at the end.

- Chaffing by GPT3: This function iterates through a data frame containing real passwords and their associated password chunks, using the 'GPT3 honeywords' function to generate honeywords for each real password.

2. API Key: The code requires an API key to access OpenAI's GPT3 model. It is essential to replace the placeholder API key with a valid one.

3. Data Source: The code appears to be using a data frame that consists of strong as well weak password chunks to generate honeywords. You might want to specify the source of this data in your thesis, as well as its significance.

4. Result Presentation: The code prepares the honeywords and their associated real passwords for presentation in the 'Chunk-GPT3 honeywords' list, which is used for further analysis and calculating HWSimilarity.

4.2.5 Honeyword Generation Methodology using Chunk-GPT3

In our honeyword generation study, we utilize the capabilities of GPT3 to create honeywords for four distinct sets of passwords, each with its unique characteristics and prompt criteria. Firstly, we have the "Strong Original Passwords" category, in which honeywords are generated based on strong, unmodified original passwords. The prompt for GPT3 instructs it to derive 19 distinct passwords that closely resemble the original password, incorporating predefined password "chunks." These honeywords are created without adding digits at the end. For this category, we initially generated honeywords for the first 100 strong, original passwords and subsequently assessed their HWSimilarity.

Moving on, we have the "Weak Original Passwords" category, which parallels the "Strong Original Passwords." Here, honeywords are produced based on weak, unmodified original passwords, utilizing the same GPT3 prompt. Similarly, we generated honeywords for the first 100 weak, original passwords and evaluated their HWSimilarity.

The third and fourth categories introduce the concept of "Strong Modified Passwords" and "Weak Modified Passwords." In these sets, honeywords are generated based on strong and weak modified passwords, respectively. To create chunks for these modified passwords, we employ a password segmentation algorithm. While the GPT3 prompt criteria remain consistent with the previous categories, the chunks are now derived from the modified passwords themselves. We have, once again, generated honeywords for the first 100 passwords in both the strong and weak modified categories and measured their HWSimilarity. This approach allows us to comprehensively assess the effectiveness of honeyword generation in various password scenarios, incorporating both original and modified password data.This approach allows us to compare the effectiveness of honeywords generated from both strong and weak orig-

inal passwords and to assess the impact of password modifications on honeyword generation. The first 100 passwords have been used as a sample set for this analysis, and HWSimilarity serves as a key metric to evaluate the quality and effectiveness of the generated honeywords.

4.3 Generating Honeywords with GPT4 without Chunks

In our quest to enhance password security, we explore a unique approach using GPT4. This approach is distinct from the traditional method of utilizing password chunks, as it directly generates honeywords from real passwords.

GPT4, with its advanced natural language processing capabilities, is tasked with creating honeywords that closely resemble real passwords, all without the intermediary step of using predefined password chunks. The GPT4 model is provided with real passwords as input and is prompted to generate a set of honeywords that meet specified criteria, such as length and character constraints.

This methodology offers a streamlined approach to honeyword generation, bypassing the segmentation of passwords into predefined chunks. Instead, it leverages GPT4's linguistic prowess to craft honeywords that are contextually convincing and apt to confuse potential attackers.

By directly generating honeywords from real passwords, this approach provides a fresh perspective on password security enhancement, offering an alternative to traditional chunk-based methods. It allows for rapid and automated honeyword creation, contributing to scalable solutions for bolstering password security. Moreover, the contextually relevant honeywords generated by GPT4 add complexity to the security landscape, making it more challenging for potential attackers to distinguish between

real passwords and honeywords.

This approach aligns with the broader mission of strengthening password security and fortifying defenses against unauthorized access and cyber threats in the digital realm.

4.3.1 Honeyword Generation Methodology Using GPT4

In the GPT4 honeyword generation process, we craft a specific prompt to instruct the model to "derive 19 distinct passwords that are similar to the given real password." Additionally, we specify that the length of these derived passwords should not exceed the length of the original password, and we emphasize not adding digits to the end of the passwords. The GPT4 model is employed for this task. Honeywords are produced for both the strong and weak password sets, and while our analysis focuses on the first 100 passwords for efficiency, it extends to assess the HWSimilarity. Notably, the prompt does not involve the use of password chunks. Instead, GPT4 independently generates similar passwords based on the original passwords, streamlining the honeyword creation process and offering a comprehensive evaluation of security measures for both strong and weak passwords.

Chapter 5

Results

5.1 Evaluation

To compare the strength of the honeywords we have used the HWSimilarity metric. The HWSimilarity metric is a method used to compare and evaluate the quality and effectiveness of honeywords generated using different techniques or methods. It is a measure specifically designed to assess how well honeywords resemble real passwords and, by extension, how challenging it is for attackers to distinguish between them. Here are some key aspects and characteristics of the HWSimilarity metric:

1. Resemblance to Real Passwords: The primary goal of honeywords is to closely mimic genuine passwords, making it difficult for attackers to differentiate between the two. The HWSimilarity metric is designed to quantify how closely honeywords generated by a particular method resemble real passwords in terms of their characteristics, such as patterns, complexity, and structure.

2. Scoring Mechanism: The metric typically assigns a score or a similarity value to each honeyword generated by a method. This score reflects how well the honey-

word aligns with the attributes of actual passwords. Higher scores indicate a better resemblance, while lower scores suggest that the generated honeyword is less convincing.

3. Evaluation of Different Techniques HWSimilarity can be used to compare honeywords generated by various techniques. This allows researchers and security practitioners to determine which method is more successful in creating honeywords that closely match real passwords.

4. Customizable Parameters The specific criteria and factors used to calculate the similarity score may vary depending on the context and the goals of the evaluation. Researchers can tailor the HWSimilarity metric to suit their specific needs.

5. Quantifying Effectiveness: The HWSimilarity metric helps in quantifying the effectiveness of honeyword generation techniques. Techniques that produce honeywords with higher similarity scores are considered more successful in deceiving potential attackers.

6. Security Enhancement: The aim of using the HWSimilarity metric is to improve the security of systems. By generating honeywords that are difficult to distinguish from real passwords, the chances of early detection of data breaches are enhanced, which in turn strengthens overall cybersecurity.

It's important to note that the specific implementation and parameters of the HWSimilarity metric may vary from one study or research project to another. Researchers may customize the metric to align with their objectives and the characteristics of the passwords and systems they are working with. Overall, the HWSimilarity metric is a valuable tool for evaluating the effectiveness of honeyword generation

Original Password	Chunk-GPT3	Re-engineered Chunk-GPT3	GPT4
vitalik3104slon	vitalik3104	vital3104slon	VitalikSlon3104
	vitalikslon	vitali3104slon	SlonVitalik3104
	vitalik3104s	vitalik3104scon	vit4likSlon310

Table 5.1: Honeyword Generation Techniques

methods in enhancing password security and data breach detection.

5.2 HWSimilarity Scores

To evaluate the effectiveness of our improved honeywords, we employ the HWSimilarity metric introduced in [8]. This metric leverages the capabilities of MPNet, a pre-trained language model, to convert honeywords and passwords into vector representations. By utilizing the semantic understanding embedded within MPNet, the system calculates the cosine similarity between each honeyword vector and its corresponding genuine password vector. This approach allows for a comprehensive assessment of the semantic similarity between honeywords and real passwords, leading to a more meaningful and reliable evaluation.

Let us illustrate some examples of the honeywords generated using the Re-engineered PwdSegment technique and compare them with the honeywords produced by Chunk-GPT3 as show in table 4.1. Additionally, we compare the overall similarity scores of the weak and strong password datasets.

In our evaluation, we're analyzing the HWSimilarity scores of three sets of generated honeywords to assess their effectiveness in mimicking real passwords as show in table 4.2. These sets are Chunk-GPT3, which generates honeywords for strong original and weak original passwords with the use of password chunks, Re-engineered Chunk-GPT3, which generates honeywords for strong Re-engineered and weak Re-engineered passwords with the same chunking technique, and GPT4, which generates

Password Strength	Chunk-GPT3	Re-engineered Chunk-GPT3	GPT4
Zxcvbn-strong	0.87	0.84	0.94
Zxcvbn-weak	0.83	0.90	0.65

Table 5.2: Comparision of Chunk-GPT3 and Re-engineered Chunk-GPT3 and GPT4

honeywords for strong passwords without utilizing any password chunks. The goal is to compare these approaches and determine which one excels in creating honeywords that closely resemble real passwords, particularly focusing on the zxcvbn-weak password category.

The table below displays the HWSimilarity results for the honeywords generated by the different approaches:

Password Strength: This column indicates the strength of the passwords being evaluated, categorized as "Zxcvbn-strong" and "Zxcvbn-weak." "Zxcvbn" typically refers to a password strength estimation tool, and these categories denote the respective password strength levels.

Chunk-GPT3: This column represents the HWSimilarity score for honeywords generated using the Chunk-GPT3 approach. For "Zxcvbn-strong" passwords, the HWSimilarity score is 0.87, indicating that these honeywords closely resemble real strong passwords. For "Zxcvbn-weak" passwords, the score is 0.83, suggesting a good resemblance to real weak passwords.

Re-engineered Chunk-GPT3: This column displays the HWSimilarity score for honeywords generated with the Re-engineered Chunk-GPT3 method. For "Zxcvbn-strong" passwords, the score is 0.84, demonstrating a reasonable resemblance to strong real passwords. In the case of "Zxcvbn-weak" passwords, the score is notably higher at 0.90, indicating that this approach excels in mimicking real weak passwords.

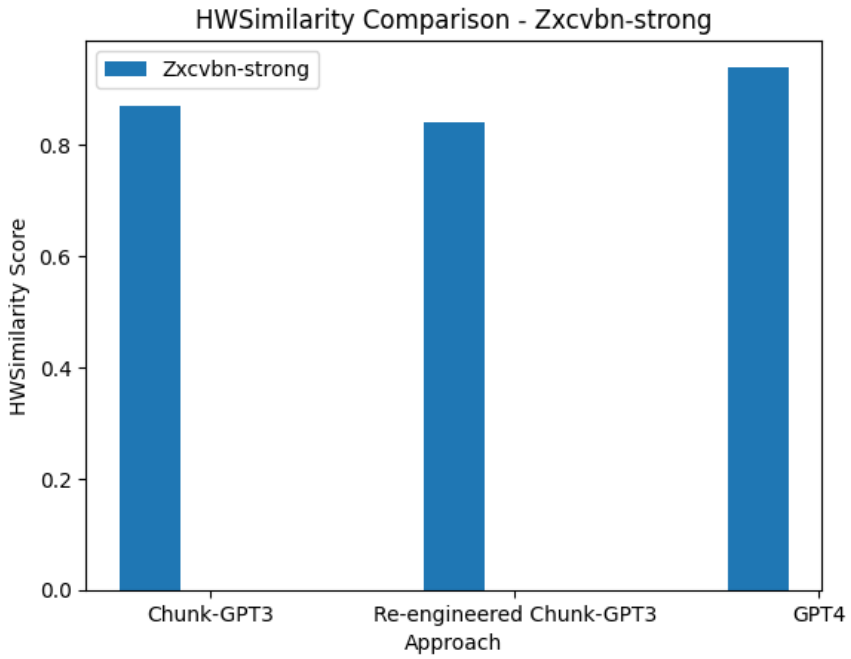

Figure 5.1: HWSimilarity Scores of Chunk-GPT3, Re-engineered Chunk-GPT3, and GPT4 for zxcvbn strong passwords.

GPT4: This column presents the HWSimilarity score for honeywords generated by GPT4 without the use of password chunks. For "Zxcvbn-strong" passwords, the score is 0.94, signifying a very close resemblance to strong passwords. However, for "Zxcvbn-weak" passwords, the score is lower at 0.65, suggesting that GPT4 may not perform as well in creating honeywords that closely mimic weak passwords.

The table illustrates that GPT4 is highly effective in generating honeywords closely resembling strong passwords, scoring 0.94 in the "Zxcvbn-strong" category. However, for "Zxcvbn-weak" passwords, its performance is comparatively weaker, scoring 0.65. In contrast, the Re-engineered Chunk-GPT3 approach excels in creating honeywords for weak passwords, with a score of 0.90, while Chunk-GPT3 performs slightly better in the "Zxcvbn-strong" category. These results suggest that the re-engineered chunking technique is particularly beneficial for zxcvbn-weak passwords.

Individual bar charts are generated for each password category (Zxcvbn-strong

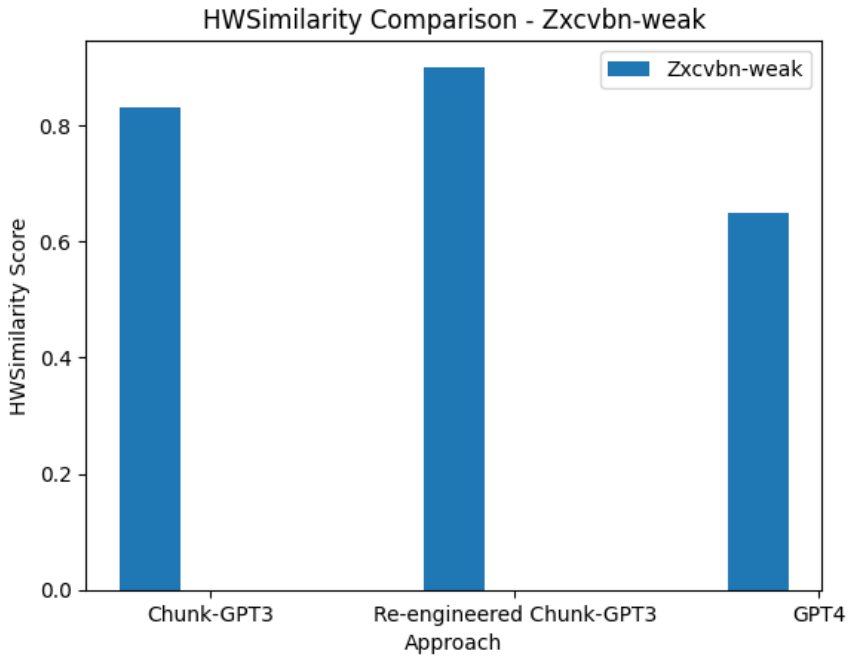

Figure 5.2: HWSimilarity Scores of Chunk-GPT3, Re-engineered Chunk-GPT3, and GPT4 for zxcvbn weak passwords.

and Zxcvbn-weak) to provide a more detailed comparison of the HWSimilarity scores for each approach. This allows for a focused assessment of how well each approach performs for a specific password category.

The generated heatmap, as shown in figure 5.3, known as the "HWSimilarity Comparative Matrix," offers a comprehensive visual representation of the comparative performance of three distinct honeyword generation approaches (Chunk-GPT3, Re-engineered Chunk-GPT3, and GPT4) across two password categories: "Zxcvbn-strong" and "Zxcvbn-weak." Each cell in the heatmap contains a numerical HWSimilarity score, reflecting how well the honeywords generated by a particular approach align with real passwords. Darker colors denote higher scores, indicating that the honeywords closely resemble real passwords, while lighter colors represent lower scores, suggesting that the honeywords may not convincingly mimic real passwords within the specific category. The annotations within each cell offer precise numerical values

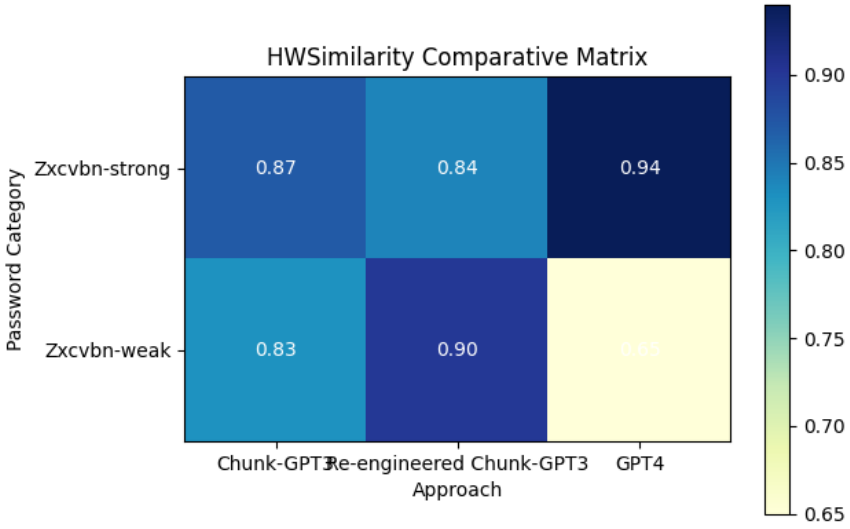

Figure 5.3: Heat map of HWSimilarity scores.

for quick reference. This visualization facilitates an in-depth analysis of the strengths and weaknesses of each approach, aiding in the identification of trends and variations in performance across different password categories, thus providing valuable insights for assessing honeyword generation techniques in enhancing password security.

Chapter 6

Future Work

When it comes to the scrutiny of passwords, it's absolutely essential to take into consideration specific patterns and data they may contain, particularly when it relates to dates of birth (DOB) and random numbers. These patterns can serve as a key insight into the potential vulnerabilities or risks associated with these passwords. Let's delve into the strategies for recognizing and effectively managing passwords that incorporate DOB information or random numbers.

Passwords that integrate DOB details, such as "12tania1995," call for meticulous examination to unveil and mitigate any associations with birth dates and years. In this context, "12" may correspond to the birth date, while "1995" signifies the birth year. To detect these types of passwords, one can conduct searches for four-digit numbers within the range of 1900 to 2100 to flag potential birth years. Additionally, in the context of dates, considering a range from 1 to 31 may be valuable. It's worth noting that single-digit dates, like "2," might also manifest as "02." Keeping a sharp eye out for these patterns is an effective means to pinpoint passwords that might contain DOB information and assess the potential risks they pose. [14]

Passwords that include DOB-related information but are expressed with a two-

digit birth year, such as "12tania95," can pose a considerable challenge. Converting these digits into alphabetic chunks can be intricate, adding an extra layer of complexity to the analysis. Passwords with two-digit birth years may not readily lend themselves to straightforward recognition or detection through conventional transformations.

When it comes to passwords featuring random numbers, as in "Tania4682," a hands-off approach is advisable. These random numbers within passwords often bear connections to personal information, like the last four digits of a phone number, a house number, or even a vehicle registration. Tampering with or altering these random numbers could inadvertently strip away valuable context or compromise the overall security of the password. Consequently, exercising vigilance and discretion when dealing with passwords containing random numbers is paramount, as they may hold significance for the individuals who use them. [2]

In certain situations, passwords with seemingly ambiguous numeric components, like "07tania," may give rise to concerns. The presence of "07" could be potentially construed as a birth date or birth month. Introducing changes or transformations to such passwords carries inherent risks, as the numeric component could bear personal significance or hint at sensitive details. Therefore, special attention should be given to the potential ramifications of modifying passwords in these particular scenarios. [13]

The analysis of passwords mandates a keen awareness of patterns tied to dates of birth and random numbers. Detecting and managing passwords containing DOB-related data necessitates a thorough assessment of birth years and dates. Passwords containing random numbers should be treated with care due to their potential associations with personal information. [14] Furthermore, passwords featuring numeric components that appear ambiguous, such as birth dates or birth months, should be subject to a nuanced evaluation while considering the possible consequences of any

modifications. [11] [5]

Looking ahead, there are several potential avenues for future research and improvement in the realm of honeyword generation and password security. One key area of focus could involve enhancing the password chunking algorithm. As observed, the HWSimilarity scores between Chunk GPT3 and Modified Chunk GPT3 do not exhibit a significant difference. Therefore, refining and fine-tuning the password chunking process may lead to an increase in the HWSimilarity score. The above-mentioned adjustments could involve optimizing the segmentation of passwords into chunks, ensuring that they effectively confound attackers while maintaining usability for legitimate users. This fine-tuning process could leverage the advanced capabilities of models like GPT4 to create more robust honeywords.

Additionally, exploring the potential of Chunk GPT4 for modified chunks in honeyword generation is a promising avenue for future work. With the continuous advancements in AI and natural language processing, GPT4 represents a cutting-edge tool that can be harnessed for improving password security. Researchers and organizations could investigate whether GPT4 can develop honeywords that outperform previous models in terms of resistance to targeted attacks. Its improved understanding of context, multilingual capabilities, and fine-tuning options may offer new strategies for generating highly secure honeywords. [12]

Moreover, it's essential to maintain a keen eye on evolving threats and attacker techniques. As attackers become more sophisticated, the field of honeyword generation must adapt to counter these emerging challenges. This necessitates ongoing research to stay ahead of potential security threats and ensure that honeywords continue to effectively protect user accounts.

In summary, the future of honeyword generation and password security holds great potential for refinement and innovation. By fine-tuning password chunking

algorithms, exploring the capabilities of advanced models like GPT4, and staying attuned to evolving cybersecurity threats, researchers and security experts can work towards more robust and effective defenses against targeted attacks and unauthorized access.

Chapter 7

Conclusions

In conclusion, passwords play a critical role in keeping our digital world secure. They are our first line of defense in online security. To make this defense stronger, we need robust methods like Honeyword Generation Techniques (HGTs) to protect our passwords from targeted attacks.

With the advancements in deep learning and natural language processing, attackers face a tougher challenge when trying to crack passwords, especially when honeywords closely resemble real passwords, even when they contain personal information.

While Chunk-GPT3 has shown its effectiveness, it has some limitations, especially in how it breaks down passwords into segments. Attackers could potentially exploit these weaknesses. Our research compared different methods of generating honeywords using GPT4, Modified Chunk-GPT3, and Chunk-GPT3. We found that GPT4 is excellent at creating honeywords that look like strong passwords but not as great with weak passwords. On the other hand, Modified Chunk-GPT3 does a fantastic job with weak passwords, while Chunk-GPT3 is slightly better for strong passwords [18].

In the end, our research highlights the strengths and weaknesses of these honey-

word generation methods, providing valuable insights into improving password security in different situations. Despite some limitations, the enhancements made to the PwdSegment technique have proven especially useful for weaker password datasets, although they might not work as well for stronger ones. Even with these minor drawbacks, honeywords remain effective as deceptive decoy passwords. They stay different from real passwords, maintain the original chunk order, and don't add numbers at the end. By introducing case variations and numerical changes, honeywords become more diverse and complex. Despite these limitations, honeywords still play a crucial role in detecting and deterring unauthorized access attempts, confusing potential attackers, and alerting administrators to potential threats. Integrating these advanced technologies into network security holds a lot of promise and significantly improves overall security measures.

Bibliography

[1] Bhadouria, A. S. Study of: Impact of Malicious Attacks and Data Breach on the Growth and Performance of the Company and Few of the World's Biggest Data Breaches. *International Journal of Scientific and Research Publications* (2022).

[2] Brown, A. Password Strength and Vulnerability Analysis: Insights and Recommendations. *Journal of Cybersecurity 15*, 2 (2020), 87–104.

[3] Chakraborty, N., Yamout, Y., and Zulkernine, M. The Tables Have Turned: GPT-3 Distinguishing Passwords from Honeywords. In *2nd IEEE Conference on Communications and Network Security - Cyber Resilience Workshop* (Florida, USA, October 2023), p. 5.

[4] Dionysiou, A., Vassiliades, V., and Athanasopoulos, E. HoneyGen: Generating Honeywords Using Representation Learning. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (2021), pp. 265–279.

[5] Grassi, P., Newton, E., Perlner, R., Regenscheid, A., Burr, W., Richer, J., Lefkovitz, N., Danker, J., Choong, Y.-Y., Greene, K., and Theofanos, M. Digital Identity Guidelines: Authentication and Life-

cycle Management. Special Publication 800-63-3, NIST (National Institute of Standards and Technology), June 22 2017.

[6] Huang, Z., Bauer, L., and Reiter, M. K. The Impact of Exposed Passwords on Honeyword Efficacy. *arXiv preprint arXiv:2309.10323* (2023).

[7] Juels, A., and Rivest, R. L. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), pp. 145–160.

[8] OpenAI. Fine-tuning, 2023.

[9] Pagar, V. R., and Pise, R. G. Strengthening Password Security through Honeyword and Honeyencryption Technique. In *2017 International Conference on Trends in Electronics and Informatics (ICEI)* (2017), IEEE, pp. 827–831.

[10] Raju, A., Filimonov, D., Tiwari, G., Lan, G., and Rastrow, A. Scalable Multi Corpora Neural Language Models for ASR. *arXiv preprint arXiv:1907.01677* (2019).

[11] Sannihith Lingutla, S. Enhancing Password Security: Advancements in Password Segmentation Technique for High-Quality Honeywords.

[12] Taylor, E. Data Analysis Methods for Password Security: A Comparative Study. *Journal of Cybersecurity Research 35*, 4 (2021), 210–228.

[13] Temoshok, D., Fenton, J., Choong, Y.-Y., Lefkovitz, N., Regenscheid, A., and Richer, J. Digital Identity Guidelines: Authentication and Lifecycle Management. Tech. rep., National Institute of Standards and Technology, 2022.

[14] Veras, R., Collins, C., and Thorpe, J. A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords. *ACM Transactions on Privacy and Security (TOPS) 24*, 3 (2021), 1–21.

[15] Wang, D., Wang, P., He, D., and Tian, Y. Birthday, Name and Bifacial-security: Understanding Passwords of Chinese Web Users. In *28th USENIX security symposium (USENIX security 19)* (2019), pp. 1537–1555.

[16] Yasser, Y. A., Sadiq, A. T., and AlHamdani, W. A Scrutiny of Honeyword Generation Methods: Remarks on Strengths and Weaknesses Points. *Cybernetics and Information Technologies 22*, 2 (2022), 3–25.

[17] Yu, F., and Martin, M. V. GNPassGAN: Improved Generative Adversarial Networks for Trawling Offline Password Guessing. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (2022), IEEE, pp. 10–18.

[18] Yu, F., and Martin, M. V. Honey, I Chunked the Passwords: Generating Semantic Honeywords Resistant to Targeted Attacks Using Pre-trained Language Models. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2023), Springer, pp. 89–108.