GUARDING THE GATE: USING HONEYWORDS TO ENHANCE AUTHENTICATION SECURITY

by

GOWTHAM KOPPADA

A capstone report submitted to the
School of Graduate and Postdoctoral
Studies in partial fulfillment of the
requirements for the degree of

Master of Information Technology Security in Artificial Intelligence

Ontario Tech University

Oshawa, Ontario, Canada

Supervisor: Dr. Miguel Vargas Martin

October 2023

CAPSTONE REVIEW INFORMATION

Submitted by: **Gowtham Koppada**

Master of Information Technology Security in **Artificial Intelligence**

Capstone Report Title: GUARDING THE GATE: USING HONEYWORDS TO ENHANCE AUTHENTICATION SECURITY

The capstone report was approved on November 17, 2023 by the following review committee:

Review Committee:

Supervisor: Dr. Miguel Vargas Martin

Second Reader: Patrick Hung

The above review committee determined that the capstone report 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

A honeyword (false password) can be defined as a duplicate password (rearranged) resembling the same characteristics of the original password. It is very challenging for any cyberpunk to distinguish between a real password and honeyword (containing PI). Using HGT's (honeyword generation technique), these honeywords are generated in lump sum and the hashed honeywords are placed in an organization database with triggers to identify breach before it's too late. In accordance with the previous research, the concept of HGT's might fail if the generated honeywords does not contain the personal information of the user, making it easy for the attacker to perform targeted attack. It is a good practice to include the chucks containing PI or part of the original password of that particular user in generated honeywords to make it look natural. Inorder to generate such honeywords with chunks, the concept of prompt engineering in LLM (Large Learning Models) is used. In this report, we tried to improve the existing prompt, making it easy for the LLM to get deep understanding and to produce better throughput. In addition to that, we compared the base GPT Learning model (existing) with the newly upgraded GPT models like GPT-3.5-turbo and GPT-4. Considering the 'strength of password' as a base factor, we came up with results and statements stating which model outperformed the others.

Keywords: prompt engineering, large learning models, honeywords, ChatGPT.

Declaration

I hereby declare that this capstone project consists of original work of which I have authored. This is a true copy of the work, including any required final revisions, as 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 the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology (Ontario Tech University) to reproduce this work by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my work may be made electronically available to the public.

GOWTHAM KOPPADA (OCTOBER 2023)

Acknowledgements

I want to start out by sincerely thanking Professor Dr. Miguel Vargas Martin. This thesis has been supported by his constant assistance, unshakable faith in my talents, and availability whenever I needed advice. I really appreciate Professor Miguel's guidance and advice for helping to shape my grasp of this topic.

I'm also very appreciative of Fangy Yu contributions to this study. Her eagerness to shed light on difficult ideas and provide insights really improved my comprehension of the topic.

Last but not least, I want to express my appreciation to my family (Dr. Koppada Rajasekhar, Amitha Raj) and friends (Alekhya Tanniru, Meher Viswanath, Navjeevan Kaur) for their consistent support and confidence in my talents, which gave me the drive to finish this report. I am really grateful for all of the help I received along the road.

GOWTHAM KOPPADA (OCTOBER 2023)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this technically evolved and globalized world, well-developed digital and network systems have increased convenience and opportunities for the human and IT industry. However, such developed digital systems have given rise to many challenges, " Security Breaches'' [12] are one of them. Therefore, growth in digital infrastructure would bring up new security breach challenges, both are directly proportional to each other.

In any security breach, passwords are the first line of defense. It is very challenging to produce a password that is tough to guess without making it impossible to remember [10]. A good password must have [1, 10]:

- Minimum length of 6-10 characters in length or 8-12 characters in length. The larger length makes it difficult to bypass.
- Must have at least three among the following: lowercase alphabet, uppercase alphabet, special character and digit. The more unique the password makes it difficult to crack.
- Alphabets, special characters, numbers and digits should be mixed up instead of just adding them in the end.

- It's a good practice to avoid using dictionary words that have common proper names.

Most visited websites "link to similarweb.com‘' and their password composition policies retrieved on May 2023 [7]:

Site	Password composition policy
facebook.com	greater than 6 characters including a letter, a symbol and a number
instagram.com	greater than 6 characters including a letter, a symbol and a number
amazon.com	greater than 6 characters
linkedin.com	greater than 6 characters
youtube.com	greater than 8 characters including a letter, a symbol and a number
office.com	greater than 8 characters of at least two types from uppercase letter, lowercase letter, symbol and number

Table 1.1: PASSWORD CRITERIA FOR MOST VISITED WEBSITES 2023

Upon achieving the above-mentioned criteria and using various hashing algorithms like SHA-256 and MD5, data can be protected to some extent. Effective strategies and solutions should be kept in place to safeguard our digital ecosystems. In case of data breach, the hacker tries various methodologies [6, 10] like hybrid attack, brute force attack,Denial-of-service Attack, dictionary attack and many more to bypass the accounts. Sometimes these data breaches can go unnoticed by the organization and can stay hidden for a prolonged period. In order to avoid such scenarios, the concept of Inserting a set of hashed honeywords among the stored passwords in password manager database is introduced. Honeywords are generated words that are similar to the original passwords using Honeyword Generation Technique (HGT) schemes. Introduction of honeywords as a security mechanism was first brought into light by Jules and Rivest [8] as a variable method of detecting password breaches. When a cyberpunk tries to hit the webpage that has embedded honeywords, with the stolen user-id and honeyword as passcode, the honeychecker would detect the anomaly and trigger an alert to the respective administrator. This HGT is one among many strate-

gies that researchers have come up with, to know if there was a password data breach in no-time. This will enable organizations to avoid further casualties. Current research has proved that involvement of AI (Artificial Intelligence) would get the work done faster and assist in getting a better throughput [2]. Using LLM(Large Learning Model), a type of AI which would understand and result in generating human-like text, has proved to give better results in producing honeywords. Here are a few examples that come under LLM's chat-gpt, DALL-E, etc. Learning models would be expecting input text-based interface (text inputs). So, here are some familiar ways to communicate with a LLM: text chat, API integration, voice assistants, text-based platforms, etc. We took the liberty to analyze the point of communication with the ChatGPT learning model through a prompt generation of honeywords. The GPT learning model uses artificial neural networks based on transformer architecture.

1.1 Contributions

- In contrast to generating honeywords through the concept of prompt engineering with provided chunks and original passwords as input to prompt using Chunk-GPT3 model, we were able to produce honeywords using the GPT-4 language model that are stronger and more resilient.

- In addition to that, We tested by generating honeywords using the GPT-3.5-Turbo language model, which was shown to be more effective than Chunk-GPT3 and less effective than GPT-4.

- We studied the existing prompt in Chunk-GPT3 and changed it so that LLM could better grasp it and create stronger/robust honeywords. Additionally, we cut down on the amount of tokens used only for the prompt sentence, which

made it more economical.

1.2 Dataset Used

This analysis made use of a password (used in previous research to show the difference in performance of the updated model) dataset referred to as 4iQ. It comprises a leaked compilation of numerous passwords and was first identified on dark web in december 2017. The collection comprises 1.1 billion unique emails, 463 million unique passwords and 1.4 billion email-password combinations. By an unidentified curator, duplicate email-password pairing was eliminated. The websites included on the list of leaked data are Canva, Chegg, Dropbox, LinkedIn, Yahoo!, and others. For the sake of simplicity, the suffix from each email address is removed and simply used the prefix as usernames [19].

In order to get valid passwords, we implemented an exclusion criterion for passwords that were either too short or too long. Specifically, passwords with less than 8 characters or more than 32 characters were omitted [13]. As a consequence, we obtained a total of 28,492 pairs of usernames and passwords. The majority of authentication systems do not allow the usage of small strings [11]. The strength of each password was assessed using the ‘zxcvbn‘ algorithm [16].

Two separate sets of username-password combinations were created taking ‘zxcvbn‘ password strength to test the HGT's performance. One strong set with 1000 username-passwords that have more ‘zxcvbn‘ password strength score and another set with less ‘zxcvbn‘ password strength score. We made use of these two datasets to produce chunks and then generated honeywords for the proposed HGT's.

1.3 Challenges

Targeted attacks: The primary obstacle in the creation of a honeyword generation technique (HGT) is in the development of honeywords that possess a high level of resilience against targeted attack [14]. In the context of targeted attacks, assailants use individuals personally identifiable information (PII) in order to make educated guesses about their passwords, hence increasing the probability of unauthorized access to users accounts. The issue at hand is of significant concern because of the widespread accessibility of personally identifiable information (PII) and passwords resulting from continuous data breaches. Additionally, individuals like to design easily memorable passwords by including personal information such as their names, birthdays, and related variations [14]. In the event that an assailant acquires a user's personally identifiable information (PII), it is probable that if a single word inside the user's list of passwords includes their PII, that particular word is extremely likely to be the authentic password while the other words are likely to be false [19]. For example the generated sweetwords for [username: xavier, dob: 1997-Oct-12, password:xavier@1997]; are [‘octbdy1992@1‘, ‘howlt1997@yamm‘, ‘qwert@89queen‘, ‘kasghtaa‘, ‘xavier@1997‘]. Assuming the attacker has the user PII in hand. It is obvious for him to figure out that ‘xavier@1997‘ is the password. So it is good practice for any HGT to make use of PII in generating the honeywords which would confuse the attacker to decide what the actual password is.

Roadmap of report: In the subsequent sections, this report will include a comprehensive overview of the underlying information, followed by a detailed exposition of the technique used to produce honeywords. Subsequently, the experimental results will be presented, accompanied by thorough comments. Finally, the study will conclude with a summary of the findings.

Chapter 2

Background

2.1 Technical Thoughts

In general, every time a user logs in to a system the entered password is checked against its hashed value present in the passwords hash table and upon confirmation/-validation the user is allowed to get into the application and access his data. If this so-called password manager gets breached, that would lead to data leak exposing all the private information of respective users.

In day-to-day life there are many such attacks happening around the world. Below are few attacking scenarios happening with respective to password breaches [8] [4]:

- Stolen files of password hashes: The file containing password hashes can be stolen by an adversary, who can then use offline brute-force calculation to crack several passwords. The adversary then could have more wide access to the password hash files on several systems, or on a single machine at different times.

- Easily guessable passwords: An adversary can successfully impersonate at least some users of a system by trying logins using common passwords since a sizable portion of users make password choices those are too bad.

- Visible passwords: When an adversary watches a person typing his password (shoulder-surfing) or sees their password written on a yellow sticky note on a display, they have access to the user's password. This attack is effectively mitigated by using a one-time password generator3, such as RSA's SecurID token.

- Same password for many systems or services: A person may use the same password across several platforms, making it possible for his password to be compromised on one system to also be compromised on others.

- Passwords stolen from users: By infecting endpoint devices, such as smartphones or laptops, with malware or by conducting phishing attacks on users, an adversary may discover user credentials.

- Password change compromised: The mechanism for allowing users to change or recover their passwords is defective or compromised, so an adversary can learn a user's password, or set it to a known value.

To avoid such scenarios among many strategies to protect the passwords, one out of them is the concept of honey-word checker. As discussed in the intro section the use of honeywords, they can be generated in many ways.

2.2 Honeywords Mechanism

The concept of honeywords were first introduced by Jules and Rivest [8]. The mechanism has four major entities [15, 19]: user, authentication server, honeychecker and an attacker.

Generation Mechanism:

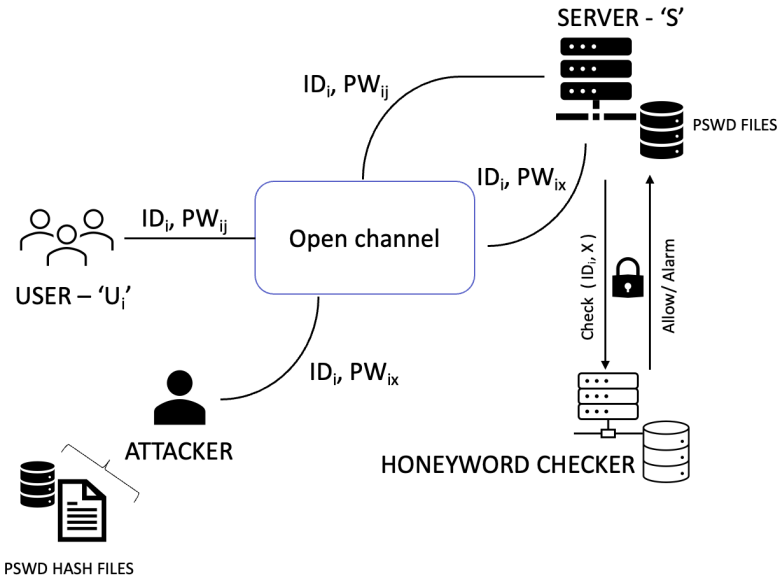

Figure 2.1: HONEYWORDS MECHANISM

- User Ui initially registered his credentials IDi, PWi.
- On the server side, the GEN(h,PWi) function call takes input of the password PWi and generates h-1 honeywords(fake passwords) using HGT. Parameter 'h' denotes the number of required honeywords.
- The generated honeywords PWij (hashed) are stored alongside the original hashed password in the DB with respective IDi.

Systems that use integrated honeywords have the capability to detect, effectively any potential security breaches. In case of breach and attacker after obtaining the database containing hashed passwords with respective salt values, the unauthorized individual (attacker) attempts to decrypt the passwords via the use of a brute force attack. Having the decrypted set ready, when an attacker attempts to compromise the server by exploiting the network using illicitly obtained hashed passwords (Ui, (IDi, PWix)). On the server side, using params IDi and PWix attempt to invoke a verification process (check(IDi, x)) with a honeyword checker. This checker examins,

internally if the provided password is authentic or a honeyword. If a honeyword is detected, appropriate measures will be implemented [8]:

- - Triggering an alarm or alerting a system administrator.
- - Allowing the login process to continue in the usual manner.
- - Enabling additional logging types of the user's activity.
- - Thoroughly investigating the origin of the login.
- - The computer system will be shut down, necessitating all users to create new passwords.

In case a valied user attempt to login with the official password, the honeyword-checker would acknowledge with an 'allow' statement.

2.2.1 Honeyword Generation Techniques (HGT's)

Honeywords generation using Tweaking:

In this Honeywords generation technique, initially after producing a set of honeywords we 'tweak' certain characters of honeywords. The character to tweak depends on the tweak position [8] t=1 or t=2. We then replace the character with the same data type, digit with digit and alphabet with alphabet, in the respective tweak position. The tweaking patterns could also be used, such as choosing the last digit position and the last special character position.

Honeywords generation using Tough-nuts:

In this particular concept by [8] discussed in [1], the system employs a set of challenging decoy passwords, referred to as honeywords, with the intention of impeding the

adversary's ability to successfully decipher the hashed password. According to the authors in reference [8], it is suggested to use this particular strategy in conjunction with another method. However, as stated in reference [6], the majority of passwords consist of basic combinations of numbers, letters, and special characters, rather than being very complex. Hence, the attacker has the capability to efficiently bypass the search for challenging obstacles. Consequently, the use of this method will effectively diminish the total level of complexity necessary to compromise the system, thereby undermining one of the fundamental objectives of honeywords, which is to augment the complexity needed to breach the system. Nevertheless, the use of "tough nuts'' as stated in reference [8] was implemented with the intention of discourage adversaries from executing dictionary attacks.

Honeywords generation using Fasttext

Dionysiou et al [5] came up with this method, which generates honeywords via representation learning. They use fasttext to turn words into vectors, and then, based on cosine similarity, they assign honeywords to the k-1 closest neighbors of a real password [19]. In the chaffing-by-fasttext approach, the use of a genuine password corpus is essential for training the fasttext model. Each word in the corpus is given a vector representation by fasttext during the training phase. Once the training process is over, the trained model may be used for querying purposes. This involves inputting a genuine password and obtaining a response in the form of a multi-dimensional vector that represents the word embedding of the given password. Subsequently, Dionysiou et al [5] used an iterative process in which they iterate over each password in their password corpus, consisting of a total of n records (where n represents the number of users). They then retrieve the k-1 passwords that exhibit the highest cosine similarity to each password, arranging them in descending order. This process results

in the generation of a comprehensive list of k × n sweetwords. In this manner, for every password included in the password file, generation of the k 1 most comparable honeywords is produced.

Honeywords generation using GPT3 learning model

The model was introduced in the year 2020 and demonstrates remarkable performance across many natural language processing tasks, including translation and question-answering. The model underwent training using an extensive corpus of text documents including billions of words.

1. Honeywords generation using Chunk-GPT3 Using GPT3 large learning model, to generate honeywords that are robust to targeted attacks by providing the semantic chunks retrieved in the PII extraction phase [19]. The quality and the diversity of the output depend on three attributes: prompt, temperature and examples given to the model.

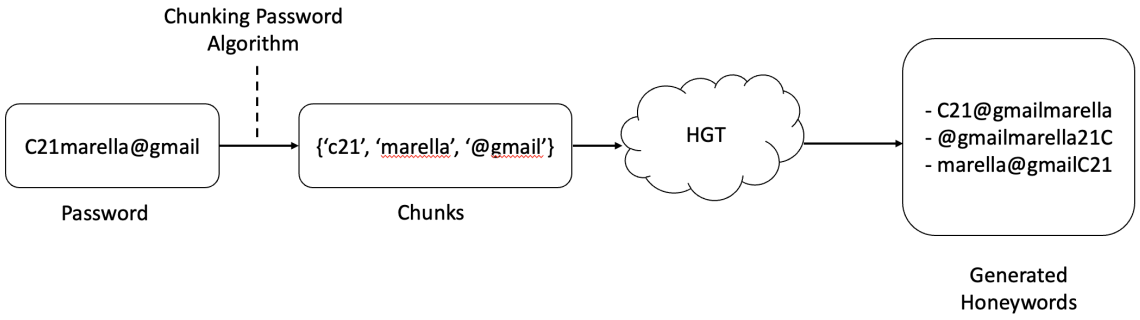

Figure 2.2: CHUNK-GPT3 FLOW

2. Honeywords generation using GPT3: Unlike Chunk-GPT3 where no input chunks are passed, here just the prompt is sent with the original password and the message [3].

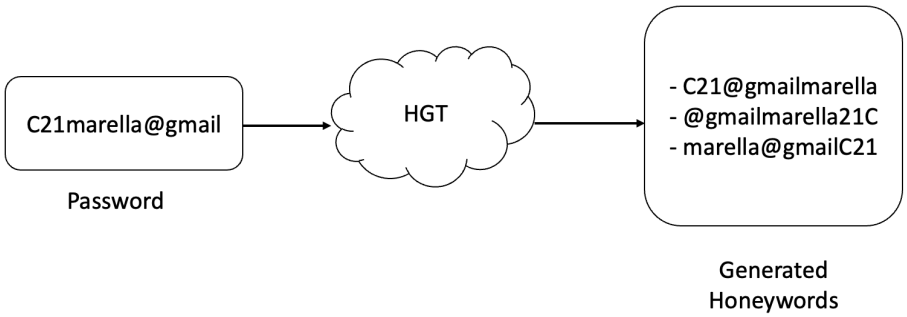

Figure 2.3: GPT3 FLOW

Prompts and Temperature are two important aspects in methods Chunk-GPT3 and GPT3. Prompts are inputs given to LLMs to procure output and Temperature is a numerical value that is in between 0 and 1 effectively regulates the model's degree of confidence when generating predictions. The temperature is a vital parameter. A lower temperature implies that the model will take fewer risks, and the honeywords created will be more repetitive while increasing the temperature results in more diversified honeywords.

2.3 Large Learning Models

Software engineers are rapidly embracing the use of large-scale language models (LLMs) [2, 20] in order to generate code and other software engineering-related artifacts. ChatGPT and GitHub Copilot are two well recognized instances of LLM-based systems used for these purposes. Based on initial research, it has been observed that chat-enabled artificial intelligence (AI) technologies had the potential to assist with several routine engineering and software development tasks [18]. There are several examples of large learning models that have been developed in recent years. Here are a few notable examples (

2.3.1 GPT Models

Generative pre-trained transformer (GPT) models have undergone training to comprehend both natural language and programming code. GPTs generate textual outputs in response to the inputs they receive. The inputs used in Generative Pre-trained Transformers (GPTs) are often referred to as "prompts‘' . The act of formulating a prompt entails the process of instructing a GPT model, often via the provision of effective job-related instructions or illustrative examples. GPTs provide the capability to fulfill a diverse array of tasks, including content generation, code development, summarization, communication, creative writing, and several other applications. Different Modes in GPT ("from link‘'):

- gpt-3.5-turbo
- text-babbage-001
- text-curie-001
- text-davinci-002
- gpt-4

The GPT-3.5-turbo model is considered the most proficient and economically efficient variant within the GPT-3.5 series. It has undergone optimization specifically for conversational tasks, while simultaneously exhibiting strong performance in traditional completion tasks. However, it is worth noting that the recent development of GPT-4, a sophisticated multimodal model capable of absorbing text inputs and generating text outputs (with the potential for picture inputs in the future), has shown superior problem-solving capabilities compared to its predecessors.

2.3.2 Prompt Engineering in GPT

Prompt engineering is the process of generating a prompt (text) that is given as input to any LLM to generate required output [20] [17]. The quality of the output(s) produced by a conversational LLM is proportional to the quality of the user prompts [9]. The provided prompts for a conversation may be used to establish interactions between a user and an LLM, hence enhancing the capacity to address a diverse range of challenges. The performance varies a lot depending on how the prompt is stated, which is a drawback of prompting.

Best Practices:

Below are few best practices, when designing a prompt :

- To get more relevant responses, it is advisable to provide specific and precise data in your prompt.
- The use of delimiters is vital in order to designate, unambiguously separate segments inside the contextual input.
- It is necessary to input a series of sequential steps with some instances. They serve as a guide, providing a clear and organized framework for the completion of the input task.
- Choose an output length that suits your needs.

Input and output text when using GPT:

Number of words used in any prompt should also be considered as an important factor when designing a prompt. Open-AI would charge for useing their API based on number of tokes used.

Prompt	"Derive three words that describes the quality of a person, characters in length between 5 to 8 for each word.''
Output	Empathy (7 characters), Resilient(9 characters), Honest (6 characters).

Table 2.1: EXAMPLE OF PROMPT IN PRODUCING RESPONSE FOR USER INPUT

Tokenization: Tokenization is a crucial procedure within the field of natural language processing (NLP) whereby text is disassembled into discrete entities, sometimes referred to as words or subwords. The aforementioned entities are often referred to as tokens. Tokenization is an essential component of natural language processing (NLP) applications, since it converts unstructured text into a structured representation that can be effectively analyzed by computational algorithms.

For example let us count number of tokens used for the statement
"Counting the number of tokens'' :
"Counting'' - 1 token,
"the'' - 1 token,
"number'' - 1 token,
"of'' - 1 token,
"tokens'' - 1 token.

The API provided by OpenAI is subject to rate constraints, which restrict the number of queries that users may make. These metrics are used to measure the performance of various models, such as tokens-per-minute, requests-per-minute (in some scenarios requests-per-day), or, in the context of picture models, images-per-minute ("from link'').

MODEL	TPM	RPM
gpt-3.5-turbo	90,000	3,500
gpt-4	10,000	200
text-davinci-002	250,000	3,000

Table 2.2: TOKENIZATION

2.3.3 ZXCVBN - Strength Calculator

'zxcvbn' is a robust, open-source password strength estimator that surpasses the rudimentary metrics typically employed in password policies. The evaluation of password strength is approached in a more sophisticated manner by a tool developed by Dropbox. This tool takes into account several aspects, such as dictionary terms, often used phrases, typing habits, and user-specific information, in order to provide a thorough evaluation of password strength. The password strength assessment tool 'zxcvbn' utilizes a scoring system to evaluate the robustness of passwords. A positive correlation exists between the score and the strength of the password, wherein an increase in the score corresponds to an increase in password strength. This resource has significant value for developers and users, since it promotes the development of stronger and more robust passwords by identifying possible vulnerabilities and providing recommendations for improvement. By integrating knowledge derived from actual instances of data breaches and methods used by hackers, 'zxcvbn' augments our capacity to protect confidential data in an ever-expanding digital landscape [19]. We make use of this function in our code base to calculate password strengths.

Chapter 3

Approach

With the research that was carried over by Fangy [19]. We have made use of the processed data from first 3 stages, to begin my analysis in stage 4 with the newer models.

Our approach in code flow consists of four stages:

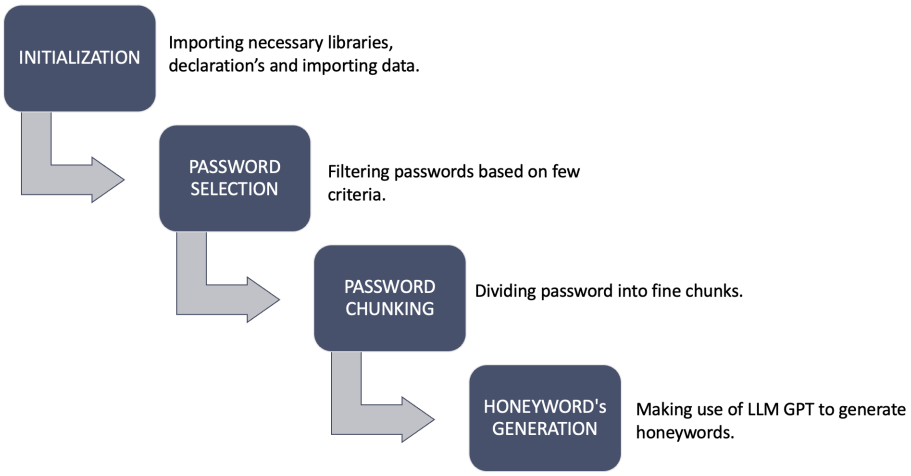

Figure 3.1: STAGES OF CODE FLOW

3.1 Initialization

The initialization step includes the establishment of all essential requirements. The necessary libraries were imported, followed by declaring global variables and then importing the password's data.

3.2 Password Selection

In the second phase, the password database undergoes a filtration process whereby passwords exceeding a length of 32 characters are eliminated. Subsequently, the passwords are classified into strong and weak categories based on their strength calculation using the 'zxcvbn' algorithm. Passwords that are deemed strong are characterized by having a strength of 4 and 3, whereas those with a strength of 2,1 and 0 are regarded weak [19]. The following table 3.1 illustrates the strengths values that are generated by the 'zxcvbn' algorithm in our code base :

Password	**Strength**
c21marella@gmail	4
Mm4Mmaxm16	3
njt1606105	2
t64t64t64t64t64t64t64t64t64t64	1
1s1s1s1s1s1s1s1s	0

Table 3.1: ZXCVBN CALCULATION

3.3 Password Chunking

In this stage we try to divide the passwords into semantic chunks and store them respectively. The PwdSegment technique is used to partition passwords into segments and segregate them both into 'zxcvbn-weak' and 'zxcvbn-strong' password sets [19].

Password	Chunks
emperorpalpateen	‘teen‘, ‘pa‘, ‘emperor‘, ‘pal‘
c21marella@gmail	‘c21‘, ‘marella‘, ‘@gmail‘
a10667007@nepwk.com	‘10667007‘, ‘.‘, ‘a‘, ‘@‘, ‘com‘, ‘nepwk‘

Table 3.2: PASSWORD CHUNKING EXAMPLES

To understand chunking table 3.2 represents examples that are generated using the ‘PwdSegmentation‘ algorithm.

3.4 Honeywords Generation

The last stage involves our proposal of using latest GPT (Generative Pre-trained Transformer) to produce honeywords that exhibit resilience against targeted assaults. This is achieved by having the semantic chunks obtained during the password chunking step, original password and the length. This involves establishing the intended functionality of the model by providing a prompt. The prompt refers to the barrier that exists between the LLM and our code base. The prompt involves receiving input values consisting of password chunks, the original password, and the length of the password. The prompt in our code base looks like:

prompt = *“Derive 19 distinct passwords that are similar to ‘’ + real password + “ and contain‘’ + chunks +“ . The length of the derived passwords should be at most ‘’ + str(len(real password)) + “. Do not add digits at the end of the passwords.‘’*

This prompt as an input would send us back the set of honeywords that were generated by large learning model and stored in DB.

Honeywords Generated when using GPT:

Prompt	"generate three distinct passwords similar to 'john123@gmail' and contain 'john', '123', '@gmail'. The length of password should be atmost 13. do not add digits at the end''
Honeywords	'123john@gmail', '123john', '@gmailjohn'

Table 3.3: EXAMPLE OF PROMPT IN PRODUCING HONEYWORDS

In this study, we are conducting experiments to generate honeywords by systematically traversing the above mentioned stages. During the last stage, there is an exploration of novel approaches to engage with a vast learning model. Therefore, for the last stage we came up with new ways of using Chat-GPT via prompt by making use of new versions like GPT-3.5 Turbo and GPT-4. Later compare the results with the existing code that uses GPT model 'text-davinci-002' [19]. In the next chapter we will be discussing about the experiment results for different models.

Chapter 4

Experiments

The first 3 phases of initialization, password selection and password chunking analysis conducted in the research study by Fangy [19] mostly remains the same. My primary goal has been on the development of phase 4 ‘honyewords‘. The following experiments are carried out to analyze different newer models of Generative Pretrained Text (GPT), alongside a comparison examination of the current model for generating honeywords.

Below are models we analyzed/experimented on in phase 4:

- Model - text-davinci-002 (existing)
- Model - gpt-3.5-turbo (experimenting)
- Model - gpt-4 (experimenting)
- Model - gpt-4 with updated prompt (experimenting)

For reference the code base (.ipnynb files) and test results are available at: https://github.com/Gowtham-Koppada/HONEYWORDS-USING-GPT4/tree/main. I used Google Colab as a computational environment for executing my code, while

PyCharm served as my integrated development environment for writing and debugging the code. The final outcomes(honeywords) of each model were copied into an Excel spreadsheet to facilitate analysis and the generation of graphs.

The graphs shown below depict the relationship between each of the user passwords and out of a total of 20 honeywords generated the amount of words with a strength of 4. We have examined the first 50 passwords from the database containing filtered data obtained from the password selection phase with a strength rating of 4, in order to conduct experimentation and analyze the models.

X-axis: represents the password of each individual user.

Y- axis: represents the number out of 20 generated honeywords having a good strength of 4.

4.0.1 text-davinci-002

text-davinci-002 is a pioneering language model that has been methodically constructed by OpenAI. The research done in the earlier study by Fangy [19] has been utilizing this model in the code. With minimal adjustments made in code, we have generated the results and made a graphical representation of data in Fig 4.1.

model =“text-davinci-002‘’

prompt =“Derive 19 distinct passwords that are similar to ‘’ + $\text{real}_password$ + $“and contain‘’ + chunks + “.The length of the derived passwords should be at most‘’ + str(len(real_password)) + “.Do not add digits at the end of the passwords.‘’$

Title: TEXT-DAVINCI-002 GENERATED HONEYWORDS WITH STRONG STRENGTH.

X-axis: Represents the password of each individual user.

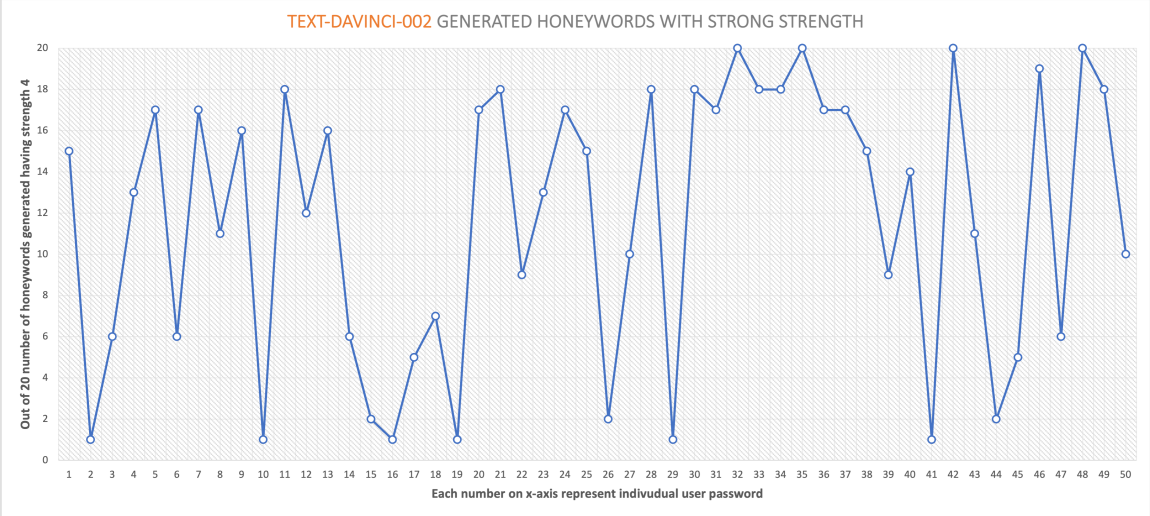

Figure 4.1: TEXT-DAVINCI-002 GENERATED HONEYWORDS WITH STRONG STRENGTH

Y- axis: Represents out of 20 generated honeywords having a good strength of 4.

Interval difference x-axis, y-axis: 1,2.

Number of Peaks reached 20: Very limited points are seen.

Trend Analysis: The graphical representation of produced results in Figure 4.1 seems crouched with many fluctuations..

Results: The honeyword generation process for this model yields a limited number of honeywords all 20 of strength 4.

4.0.2 GPT-3.5 Turbo

GPT-3.5 Turbo, similar to its predecessor GPT-3, is an advanced language model that has been created by OpenAI. The system is specifically built to perform a range of language-related activities and is known for its remarkable capacity to produce text that closely resembles human-generated texts. Using this gpt-3.5-turbo, we have developed a new model that inputs prompt with params: chunks,password length and original password. Upon generating the honeywords data we have made a pictorial

representation in the form of a graph Fig 4.2.

model =“gpt-3.5-turbo‘’,

messages =[“role‘’: “user‘’, “content‘’: “Derive 19 distinct passwords that are similar to ‘’ + real$_p$*assword* + “*andcontain*‘’ + *chunks* + “.

Thelengthofthederivedpasswordsshouldbeatmost‘’ + *str*(*len*(*real*$_p$*assword*)) + “.

Donotadddigitsattheendofthepasswords.‘’]

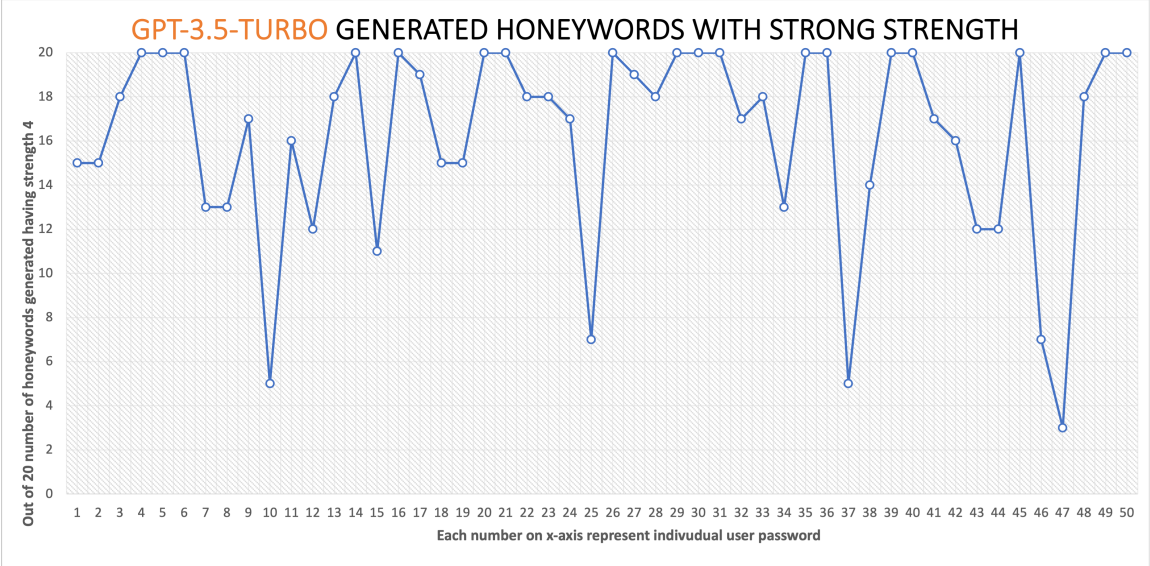

Figure 4.2: GPT-3.5-TURBO GENERATED HONEYWORDS WITH STRONG STRENGTH

Title: GPT-3.5-TURBO GENERATED HONEYWORDS WITH STRONG STRENGTH.

X-axis: Represents the password of each individual user.

Y- axis: Represents out of 20 generated honeywords having a good strength of 4.

Interval difference x-axis, y-axis: 1,2.

Number of Peaks reached 20: Decent amount of points are seen.

Trend Analysis: The graphical representation of produced results in Figure 4.2 seems to have upward and downward trends.

Results: The experiment results came out to be better. There is an upward trend

when compared to the existing test-davinci-002 model in Fig 4.1.

4.0.3 GPT-4

GPT-4 signifies the further advancement in OpenAI's lineage of language models, further enhancing the accomplishments of its forerunners. Although there may be variations in the particular architectural design and features, it is expected that GPT-4 will uphold the trajectory of advancing the capabilities of AI language models. We took advantage of this and developed a new model that inputs prompt with params: chunks, password length and original password. Generated graph Fig 4.3 using the produced data using GPT-4 model.

model =“gpt-4“,

messages =[“role“: “user“, “content“: “Derive 19 distinct passwords that are similar to “ + real$_p$*assword* + “*andcontain*“ + *chunks* + “. *Thelengthofthederivedpasswordsshouldbeatmost*“ + *str*(*len*(*real*$_p$*assword*)) + “. *Donotadddigitsattheendofthepasswords.*“]

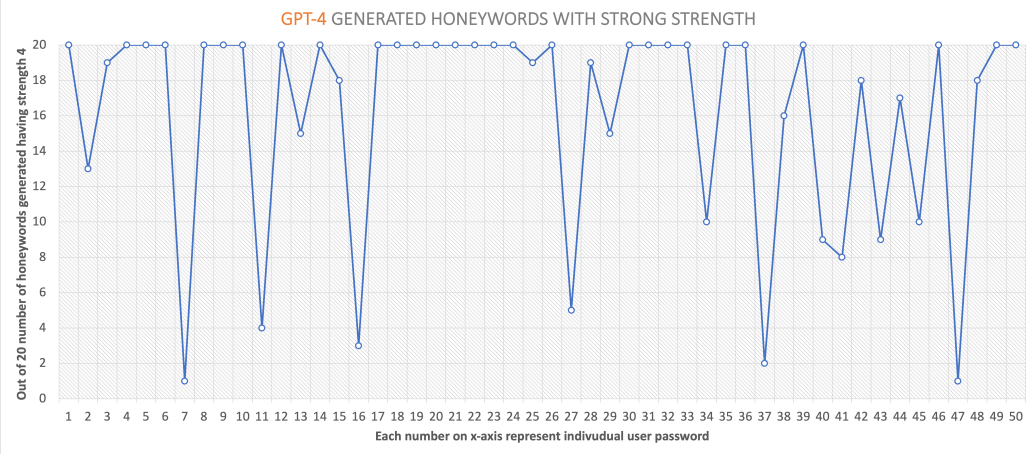

Figure 4.3: GPT-4 GENERATED HONEYWORDS WITH STRONG STRENGTH

Title: GPT-4 GENERATED HONEYWORDS WITH STRONG STRENGTH.

X-axis: Represents the password of each individual user.

Y- axis: Represents out of 20 generated honeywords having a good strength of 4.

Interval difference x-axis, y-axis: 1,2.

Number of Peaks reached 20: fine amount of points are seen, with a linear trend.

Trend Analysis: The graphical representation of produced results in Figure 4.3 seems to have less downward trends and show few linear trends.

Results: The experiment results came out to be better. There is a very good improvement when compared to the existing text-davinci-002 model.

4.0.4 GPT-4 with updated prompt

The prompt has been revised to enhance comprehension for the LLM, resulting in the generation of more resilient output compared to earlier outcomes. Furthermore, there has been a little decrease in the quantity of tokens used while just evaluating the words in the prompts, resulting in improved cost efficiency compared to previous levels. Upon running the gpt-4 model with updated prompt we have acquired the date and generated the graph Fig 4.4.

Old Prompt: "Derive 19 distinct passwords that are similar to '' + real password + " and contain'' + chunks + " .The length of the derived passwords
should be at most '' + str(len(real password)) + ". Do not add digits at the end of the passwords.''

New Prompt: "Generate 19 distinct passwords resembling '' + real password + ",including '' + chunks + ", with a maximum length of '' + str(len(real password)) + "characters and no added digits at the end.''

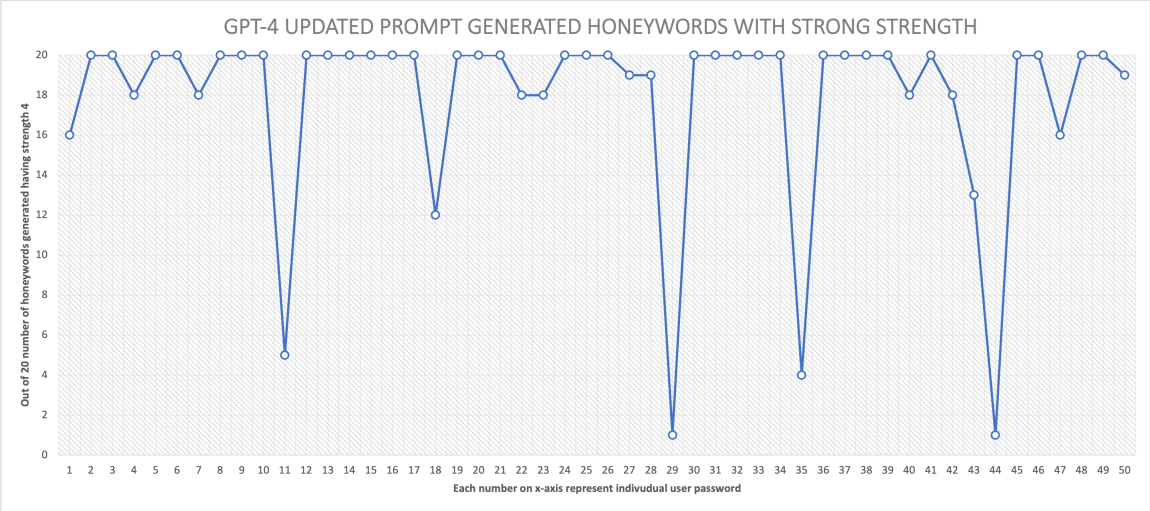

Figure 4.4: GPT-4 UPDATED PROMPT GENERATED HONEYWORDS WITH STRONG STRENGTH

Title: GPT-4 UPDATED PROMPT GENERATED HONEYWORDS WITH STRONG STRENGTH.

X-axis: Represents the password of each individual user.

Y- axis: Represents out of 20 generated honeywords having a good strength of 4.

Interval difference x-axis, y-axis: 1,2.

Number of Peaks reached 20: A very good amount of points are seen, with a linear trend and less downward trends.

Trend Analysis: The graphical representation of produced results in Figure 4.4 seems to show more linear trends and less downward trends.

Results: The experiment results came out to be better. Proving outstanding performance when compared to the existing text-davinci-002 model.

Chapter 5

Discussion

5.1 Statistics

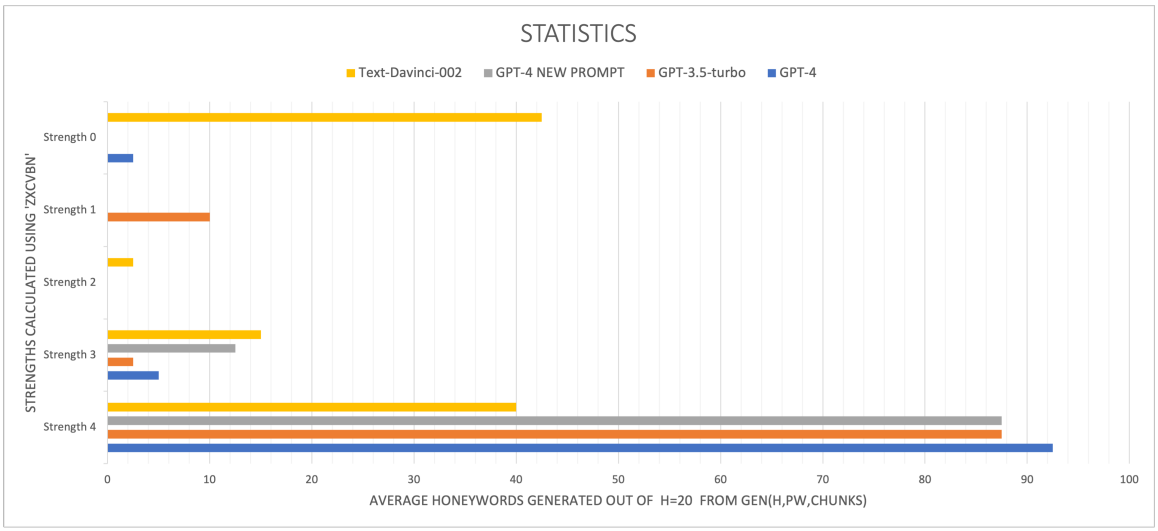

Figure 5.1: GPT MODELS STATS OF DATA WITH STRENGTHS [4,3,2,1,0]

X-Axis: percentage of honeywords on an average generated out of 20.

Y-Axis: Strength [4,3,2,1,0] values.

When doing a comparative analysis between the preexisting model ‘test-davinci-002‘ and the recently generated models ‘gpt-3.5-turbo‘ and ‘gpt-4‘, taking strength as base

factor. We came up with the following deductions. When considering the strengths provided by the ‘zxcvbn‘ function, it is important to focus just on those with values of 4 and 3, since these values have been shown to indicate strong passwords. Strengths 2, 1, and 0 are regarded as lacking in robustness. The model ‘test-davinci-002‘ has produced 40% of passcodes with a strength of 4 and 15% with a strength of 3. When comparing it to gpt-3.5-turbo, the system generated honeywords with a strength of 4 at an accuracy rate of 87.5%, whereas honeywords with a strength of 3 were generated at a rate of 2.5%. Enhancing its model superiority relative to the preexisting ‘test-davinci-002‘ model. When comparing the capabilities of GPT-4 and GPT-3.5-turbo, it is seen that GPT-4 has successfully created 92.5% of passwords with a strength rating of 4, while 5% of the passwords generated have a strength rating of 3. Demonstrating superior efficacy compared to other models.

	ST 4	ST 3	ST 2	ST 1	ST 0
text-davinci-002	40%	15%	2.5%	0%	42.5%
gpt-3.5-turbo	87.5%	2.5%	0%	10%	0%
gpt4	92.5%	5%	0%	0%	2.5%
gpt-4 updt	87.5%	12.5%	0%	0%	0%

Table 5.1: EXPERIMENTED RESULTS STATISTICS

Subsequently, we have considered the very effective GPT-4 model and attempted to modify the prompt. The newly devised prompt has yielded superior outcomes compared to the current paradigm. Despite the fact that the revised prompt gpt-4 has yielded a mere 87.5% of words classified as strength 4, it has created an additional 12.5% classified as strength 3, resulting in a cumulative total of 100%. No honeywords were produced with strengths 2, 1, and 0. Demonstrating its increased efficiency. Upon Obtaining all the results we have generated a final combined analysis of data for all models with strengths [4,3,2,1,0] in Fig 5.1.

5.2 Limitations/Future work

- During the course of my investigation, it was found that the generation of honeywords for numeric passcodes deviates somewhat from the intended path. For example: For password ‘5,131,424,096‘ HGT is considering ‘,‘ as one of the chunks. As a result, honeywords [‘945,,314,480,683,060‘, ‘314,480,683,,,‘] are generated, including consecutive commas. This issue of chunking should be addressed in future research.

- Furthermore, it is worth considering the potential for further investigation into prompt engineering as a means of enhancing the efficacy of prompt generation for emerging GPT models in the production of honeywords.

Chapter 6

Conclusions

The use of expansive learning models has facilitated the development of strong honeywords in a timely manner, hence simplifying our task. Recent advancements in natural language processing have yielded new models, such as GPT-3 and GPT-4, which have shown enhanced effectiveness in comparison to their predecessors. This study presents an analysis and the corresponding data, which provide evidence that GPT-4 exhibits superior performance in comparison to its predecessors, namely GPT-3.5-turbo and text-davinci-002. In the future, we suggest using GPT-4 as an enhanced Honeyword Generation Technique in my proposed paradigm.

Bibliography

[1] Akshima, Chang, D., Goel, A., Mishra, S., and Sanadhya, S. K. Generation of secure and reliable honeywords, preventing false detection. *IEEE Transactions on Dependable and Secure Computing 16*, 5 (2019), 757–769.

[2] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. advances in neural information processing systems. https://doi.org/10.48550/arxiv.2005.14165, 2020.

[3] Chakraborty, N., Yamout, Y., and Zulkernine, M. The tables have turned: GPT-3 distinguishing passwords from honeywords. In *2023 IEEE Conference on Communications and Network Security (CNS)* (2023), pp. 1–5.

[4] Dastane, O., Bakon, K., and Johari, Z. The effect of bad password habits on personal data breach. *International Journal of Emerging Trends in Engineering Research 8* (10 2020), 6950–6960.

[5] Dionysiou, A., Vassiliades, V., and Athanasopoulos, E. Honeygen: Generating honeywords using representation learning. *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (2021).

[6] Erguler, I. Achieving flatness: Selecting the honeywords from existing user passwords. *IEEE Transactions on Dependable and Secure Computing 13*, 2 (2016), 284–295.

[7] Huang, Z., Bauer, L., and Reiter, M. K. The impact of exposed passwords on honeyword efficacy. arxiv preprint arxiv:2309.10323, 2023.

[8] Juels, A., and Rivest, R. L. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, Association for Computing Machinery, p. 145–160.

[9] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv. 55*, 9 (jan 2023).

[10] Summers, W. C., and Bosworth, E. Password policy: the good, the bad, and the ugly. In *Proceedings of the winter international synposium on Information and communication technologies* (2004), pp. 1–6.

[11] Tan, J., Bauer, L., Christin, N., and Cranor, L. F. Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2020), CCS '20, Association for Computing Machinery, p. 1407–1426.

[12] Thomas, K., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., Markov, Y., Comanescu, O., Eranti, V., Moscicki, A., Margolis, D., Paxson, V., and Bursztein, E., Eds. *Data breaches, phishing, or malware? Understanding the risks of stolen credentials. Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (2017).

[13] 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)* (Santa Clara, CA, Aug. 2019), USENIX Association, pp. 1537–1555.

[14] Wang, D., Zhang, Z., Wang, P., Yan, J., and Huang, X. Targeted online password guessing: An underestimated threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, Association for Computing Machinery, p. 1242–1254.

[15] Wang, D., Zou, Y., Dong, Q., Song, Y., and Huang, X. How to attack and generate honeywords. In *2022 IEEE Symposium on Security and Privacy (SP)* (2022), pp. 966–983.

[16] Wheeler, D. L. Zxcvbn: Low-budget password strength estimation. In *Proceedings of the 25th USENIX Conference on Security Symposium* (USA, 2016), SEC'16, USENIX Association, p. 157–173.

[17] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. A prompt pattern catalog to enhance prompt engineering with ChatGPT. arxiv preprint arxiv:2302.11382, 2023.

[18] White, J., Hays, S., Fu, Q., Spencer-Smith, J., and Schmidt, D. C. ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arxiv preprint arxiv:2303.07839, 2023.

[19] Yu, F., and Martin, M. V. Honey, i chunked the passwords: Generating semantic honeywords resistant to targeted attacks using pre-trained language models. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 20th International Conference, DIMVA 2023, Hamburg, Germany, July 12–14, 2023, Proceedings* (Berlin, Heidelberg, 2023), Springer-Verlag, p. 89–108.

[20] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large language models are human-level prompt engineers. arxiv preprint arxiv:2211.01910, 2023.