An Object Detection Based Implementation of Password Generation for Strength and Memorability

Alexander Lotero
aal562@nyu.edu

Abstract---Password Policy remains one of the most prominent discussions in cyber security. On paper, password policy is easy: require that users deploy what your system deems a strong password. However, in practice, it is much more complicated than that. Users must strike a balance between strength and memorability. Utilizing the ImageAI Python library, we introduce and evaluate the viability of machine learning object detection for password generation and memory. We deploy a Python program that implements ImageAI, partnered with a library to download a random image from Google images. Objects are identified within the image and put together to create a password that hopes to mimic the Correct Battery Horse Staple method of password generation. This image is then provided to users to serve as a reminder of their passphrase for future logins. Our goal is to both generate what password strength meters consider a strong password, and relieve users of the full responsibility of remembering by providing an image to supplement their memory. Our preliminary results show promise for the benefits of images as password reminders, but the passwords generated suffer from the simplicity of our early model.

I. Introduction and Motivation

Perhaps the most public facing domain within cybersecurity is passwords and how we make them. This domain encompasses everything relating to passwords including password creation policy, measurements of strength - length and characters, and maintaining security throughout the life of the password.

Password strength has been found to be a tricky problem to "solve." This remains the case due, in part, to the inherent tradeoff that users face between strength and memory. Creating a strong password is not fundamentally difficult. A user can randomly type a 16 character string that includes a combination of upper-case letter, lower-case letters, numbers, and special characters and they have created what most password strength meters consider a very strong password. The difficulty comes later when users are required to remember those 16 random characters: the characters used and in what order. This difficulty grows exponentially when considering that, according to a research by NordPass in 2020, the average user has 100 passwords [1]. When considering 100 passwords, it is all but impossible to imagine the average user selecting what can be considered truly strong passwords for each account. Thus, it becomes a balancing act between strength and being sufficiently memorable.

This balance, and the difficulty that comes with perfecting it, leads many users to rely on things about themselves such as names, birth dates, favorite sports teams, etc [2]. Users are making the decision to ignore security and favor memorable passwords instead. This makes the job of malicious actors much easier than it should be (see Figure 1). In order to address this, we evaluate this balance and its components.

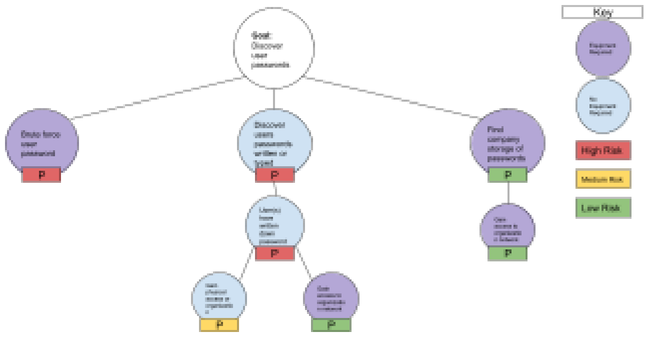

Goal: Discover user passwords
Brute force user password
Discover users passwords written or typed
Find company storage of passwords
Key
High Risk
Medium Risk
Low Risk
P

Fig. 1: *Attack tree for malicious actors attempting to compromise passphrases. Note the assignments of "Equipment Required" vs. "No Equipment Required" and the risk classification of each task.*

This paper is concerned with both aspects of the balance: strength and ease of memory. Our approach to the strength aspect of the problem is to implement the Correct Horse Battery Staple method [3]. The Correct Horse Battery Staple method is an idea in cybersecurity in which a password composed of four random words, even of only lowercase letters, is stronger than a shorter password that includes a wider variety of characters. To apply this method, we utilize a program with the ImageAI Python library designed for image recognition. The program takes an image as input (.png or .jpg), detects 4 objects within the image, randomizes the order of the names of the objects, and finally, provides a string of 12+ lowercase letters (a password).

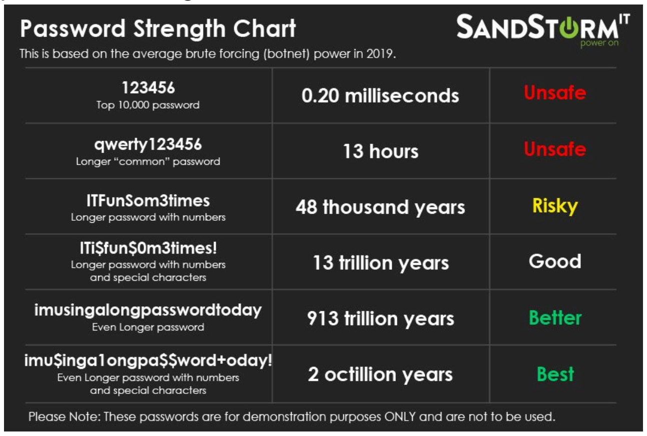
In addition to creating what we hope can be considered a suitably “strong” password, our implementation hopes to address the issue users face with remembering their passwords [4]. After the user is provided with their passwords, we propose permitting users to save a copy of the image used to generate said password. This image can serve as a reminder/hint of what their password was. We theorize that a sufficiently “busy” image will be both an adequate hint for the user, and be acceptably difficult to guess even if the image is compromised.

II. Hypothesis

A program that implements the Correct Horse Battery Staple password method through image recognition techniques will generate long, and thus strong random passwords that take significant computing power and an unfeasible amount of time to brute force. Previous works have attempted to combine images and password generation through the use of user drawings [5]. This method retains a weakness of traditional user selected passwords, it is still coming straight from the users mind. Knowledge of the user can provide insight for attackers to uncover a password generated by this method. Our method severs this user-password relationship. By utilizing a random image, knowledge of the user, their interests, hobbies, etc. will have no bearing on their password. Additionally, our project proposes allowing users to store their image as a “hint” of sorts to assist their memory and discourage written reminders of their passwords. A perceived weakness in memory can influence users into selecting a weak, easy-to-remember password [6]. We shall overcome this obstacle and create a stronger password generation method by minimizing the role of memory in password creation and recall.

III. Metric

The metric we shall use to evaluate our evidence, because we are dealing in passwords, is *time to brute-force*. Meaning, how long might it take an adversary to crack a password, likely using dedicated hardware. For example, SandStorm IT [7] has a graphic comparing different “types” of passwords employing different methods for increased strength (see Figure 2). Several of these passwords’ time to brute-force deal in the trillions of years, as such times will be estimated by several online password strength meters.



Password Strength Chart

This is based on the average brute forcing (botnet) power in 2019.

123456 Top 10,000 password	0.20 milliseconds	Unsafe
qwerty123456 Longer “common” password	13 hours	Unsafe
ITFunSom3times Longer password with numbers	48 thousand years	Risky
ITi$fun$0m3times! Longer password with numbers and special characters	13 trillion years	Good
imusingalongpasswordtoday Even Longer password	913 trillion years	Better
imu$inga1ongpa$$word+oday! Even Longer password with numbers and special characters	2 octillion years	Best

Please Note: These passwords are for demonstration purposes ONLY and are not to be used.

Fig. 2: *Password strength chart based on average brute-forcing power in 2019.*

IV. Related Research

Passwords and password policy remain two of the hottest topics in cybersecurity because of their presence in every computing device around the world. As such, countless researchers have approached the domain with what they believe to be a “fix” for one aspect or another within the domain. Jermyn et al. [9] and Guo et al. [5] adopted strategies to utilize the visual sense to assist in engaging the user’s memory from the time of password creation onward. In Guo’s work, *Optiwords*, they employ a draw to text method in which users draw “shapes" over the image of a keyboard. Those drawings are then translated to key-presses and turned into a password. They reported little-to-no increase in memorability when compared to existing password policies. That said, other studies [10, 11] draw on evidence in psychological research that the human brain more easily remembers images than text characters.

Further explorations of graphical passwords from Chiasson et al. [12] examine a similar approach to ours of utilizing existing images, but they developed *Cued Click Points*. In their work, when a user requests to be authenticated, they are provided with a set of images, the same set that they were provided at the time of password creation. Correct password entry, comparable to entering the proper string in a traditional password entry implementation, is done by clicking the proper point(s) within the image. They target the same memorability found within images that we do. Despite having the added benefit of image memory over text memory, their implementation retains a similar “burden of memory” that is found within traditional text-based passwords. Additionally, remember that this burden grows in complexity with the number of passwords that the user must remember.

An expanded study by Meng [13] attempts to scale graphical password theory to the many passwords dilemma. In his *RouteMap* project, he has users "draw a route on a map as their passwords," (note the plural passwords). This was his attempt to scale implementations of graphic passwords like *Cued Click Points* [12] to users with many passwords. Meng reports increased memory performance when evaluated on users with multiple passwords. Our model attempts to address this same scaled memory problem, but each image is entirely independent of every other image. Users are provided with an image for every password, rather than a map that attempts to benefit memory for all passwords combined. Our model intends to address the weakness, or perceived weakness [4], in human memory, while keeping passwords, and all related material independent of one another.

V. Empirical Evidence

Our evidence section is divided into several subsections to provide what we believe to be the most thorough and understandable explanation. This includes an explanation of our model, a peek inside our code, and a detailed analysis of our findings.

A. *Assumptions*

Our model features two important assumptions. Our first assumption deals with concerns beyond the scope of this project. We assume that our program's process of downloading an image from Google, utilizing the google_images_download module for Python, is completed in a secure manner. While not revealing the password completely, if an attacker were to intercept the image in transit, they have access to the same hint that is intended for the user for later password memory. This image is designed to help users remember their generated password, but in-turn, it also provides malicious actors with an advantage in guessing said password. As such, for the purpose of this research, we assume the download is completed securely with no interception.

Our other assumption relates to organizations' existing password policies. Our hope is that our program and password generation process can be applied anywhere that a password is needed, from individuals who need a password for a streaming service, to entire organizations that deploy a universal password creation policy. That said, in its current state, our program produces passwords between 10 and 20 characters in length, with the option to identify more objects in images and subsequently generate longer passwords. Some existing password policies, whether to ensure legacy support or some other reason, still maintain password length limits as low as 16 characters. As such, we assume adoption of our technique is preceded by the inclusion of a generous password length limit.

B. *The Model*

With this project, we hoped to address two essential aspects/issues within password policy: password strength, and user memory. As such our model operates as follows:

1. A user creating a new account or changing the password of an existing account opens our program to generate a new password.
2. The program runs and takes no input from the user[1].
3. Utilizing the existing *google_images_download* script within Python [14], a random image is found and downloaded.
4. This newly downloaded image is then run through the ImageAI library for object detection.
5. A list is created containing strings of every object identified within the image.
6. The list is then checked for duplicates, and deleted if found.
7. The list is then checked to confirm that at least three unique objects were identified, if not, step 3 onward is repeated with a new image until the object minimum requirement is satisfied[2].
8. Next, the strings of identified objects are combined to create a single passphrase with all spaces removed.
9. This passphrase is then printed to the user for usage in the new account creation.
10. Finally, the image is provided to the user for download and storage for later use.

C. Pseudo Code

For a better understanding of the process, examine the following logic utilized:

```
Call downloadImages() to download a
      random image from the google
      images site
Call generatePassword() with the image
      as its sole argument
Create a Python list "detections" of the
      objects identified within the
      image
Create an additional Python list
      "objects" to copy the objects
      into, minus any duplicates
Loop iterating through the original
      objects list:
```

[1] We have considered an alternative in which users provide the program with an image of their choosing.
[2] This is to aid in our hope that Google is providing busy images.

```
If current entry is not in new list,
      then add it
Otherwise, continue
Create "password" string by joining all
      entries from the "objects" list
If there are any spaces (" ") within
      "password," remove them
Print "password" to the user
```

D. Results

Our results are twofold, both of which appear in a qualitative state. First, let's examine a password that our program generated and begin to understand how well the program generates what could be considered a strong password. The ImageAI function output of our sample image of a generic city is provided (see Figure 3).

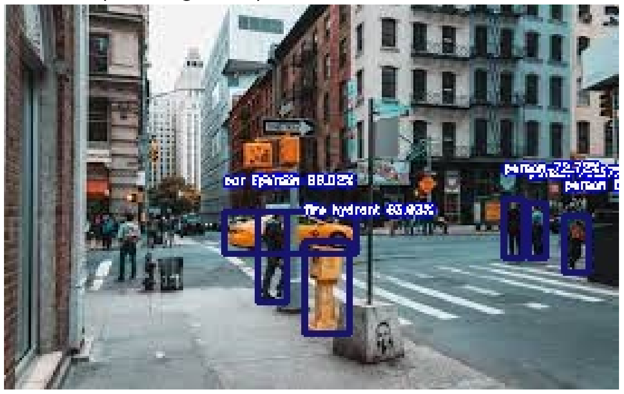


Fig. 3: *Output of our ImageAI function with the generic city street image as input. Note: The new image with the objects outlined is never provided to the user, only the original image is their hint.*

As you can see, the function identified a car, a fire hydrant, and several cases of a person. However, the several cases of persons were reduced to one by our loop to eliminate duplicates. Thus, our final resulting password was: "personcarfirehydrant." We then measure our newly found demonstration password's strength [15, 16, 17]. The first meter, provided by *howsecureismypassword.net*, reports a time to crack of 15 billion years, so what we consider very strong. The next meter, from *my1login.com*, labels this password "strong" with a time to crack of 10 months. The final meter, a resource from BetterBuys, has a time to crack of infiniti, meaning the password's strength is beyond the scope of nearly all adversaries' resources. So, while all three meters agree that "personcarfirehydrant" is at the least a strong password, *my1login.com* considers it crackable in as little as ten months, while the other two have it as just about impossible (see Figure 4). This difference can be explained by the methods used by these meters. Howsecureismypassword and BetterBuys appear to solely envision the time it would take to brute force based on character variety or length, strong emphasis on "or." my1login appears to judge based on variety and length together, so if it is not very strong in both departments, then it is not very strong altogether. We came to this conclusion by changing a single character to uppercase which increases our time-to-crack to 2 years. While these strength measurement results are promising, this password remains invalid for any policy that requires a certain variety of characters including a mix of uppercase, lowercase, and numbers. Additionally, this image and subsequent password only represent the ideal case for our hypothesis, but more on that later.

Next, let us consider the other half of our hypothesis: user memory. While our study was not performed on a large, significantly diverse sample, we believe the implications to be an excellent sign and incentive to perform further investigation. Our tests serve as a proof-of-concept. We provided 11 individuals with a generated password and the original image used to create it. The individuals were then periodically asked over a two-and-a-half week period to recall their password, first without using the image and then with the image as a hint. Of our small 11-person sample, every individual reported that their image was a sufficient hint to help them remember their passwords, particularly at first. Further, several individuals reported that after 3-4 times being asked for their password and looking at the image in order to remember, they no longer needed the image. The addition of an image to look at early in the password's life provided the users with an additional memorization resource. As referenced earlier, human's find it easier to recall images than words, as such, instead of memorizing the actual password, our participants were memorizing their hint image and recalling that in later trials in order to remember their password. We consider this to be promising early evidence of integrating image hints into password policy.

E. Analysis of Results

While we do consider the results of our preliminary work to be promising, in the previous section we alluded to the concept of our "ideal case." The password generation process itself was found to be far from perfect. Primarily, we encountered problems implementing the *google_images_download* code into our program. The implementation remains somewhat incomplete/buggy, and as such cannot be relied upon to properly fetch a random image. Completion of this implementation requires more time and work beyond the scope of this project. As such, to test the remaining aspects of our project, we were forced to switch to an online random image fetcher to collect an image to input into the object detector.

Moving on to cases in which we input an image, before explicitly coding the 3 object minimum within our program, we unleashed the ImageAI on several images, only to receive a majority of cases in which the model only identified one or two objects. It wasn't until we implemented the minimum that the fatal flaw within our model became obvious. The ImageAI model, in its current state, with the neural network's current training, does not have a sufficiently large vocabulary nor a proper ability to identify less pronounced objects within images. These two shortcomings prohibit our model from reaching what we consider to be a threshold for "uniqueness" within the password generation process.

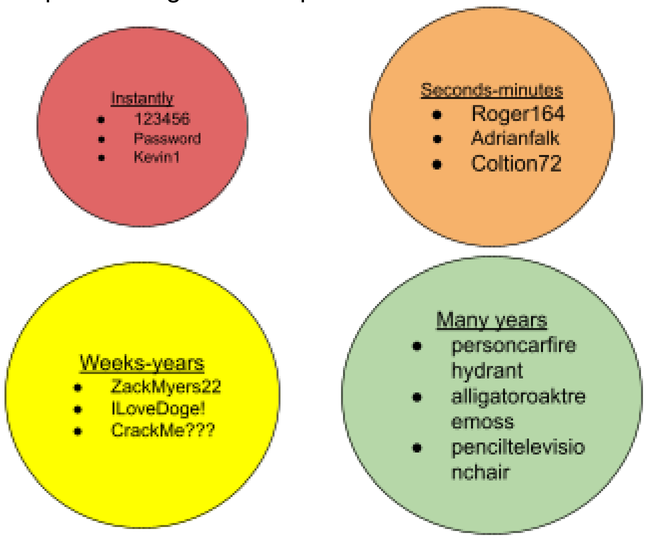


Fig. 4: *Sorting different passwords into the time-to-crack according to howsecureismypassword. Weak passwords pulled from a list of 200 most common passwords.*

Consider a case in which two unrelated users employ our program to generate unique passphrases. An image for each is selected at random. One user's image is of a man exploring a swamp in Louisiana, the other is of an elementary school musical production. On paper, these images seem unique enough from one another. These images are then processed by our model. The ImageAI detector is almost certainly going to identify a person within each image, potentially a tree from the swamp and a tree serving as a prop in the musical, and as little as one more object. The users are given the following two passwords: "persontreeboat" and "persontreechair." We introduce this worst case example to demonstrate the shortcoming of the model's vocabulary. The passwords generated are not suitably unique if every landscape image produces a passphrase that includes "tree" or every street image produces a phrase with "person."

Instead, consider a model with expanded training that results in a more diverse vocabulary of objects. The same example images from above might produce phrases like "swampmosquitobaldcypresstree" and "stageactormicrophone." We would credit these outputs as being more "independent" of one another than the previous hypothetical output.

This example illustrates why we believe that despite the drawbacks of our model in its current state, and our imperfect hypothesis, our results demonstrate the possibilities with a more advanced model for the passphrase generation step. Further, our results begin to validate our claim that an image can serve as a sufficient resource to aid the human memory. We expand on these ideas in our upcoming Future Work section.

VI. Future Work

We consider our findings to be a promising proof-of-concept, but we believe the real breakthroughs await in later iterations. Work for the immediate future is to get our implementation of *google_images_download* running properly within our program to fetch images. This is our essential first step to ensure that all aspects of our project operate within our program.

Following a completed program, our next step would be to test our current model on a much larger, diverse sample to re-enforce any conclusions that we have drawn. For example, we believe the biggest strength of our current model is its ability to supplement human memory. Our findings at this stage demonstrate that potential, but to draw any proper conclusions, we'd first need results from a truly representative sample.

Beyond our current model, we believe future work should address the obstacles found in our current implementation. For starters, we discussed how the biggest obstruction, as of now, is the simplicity of our model. The solution could come in one of two forms. Firstly, we could spend additional time training the ImageAI neural network to identify smaller, less pronounced objects within an image e.g. a dragonfly roaming in the background of a graduation photo. Alternatively, we may benefit from adopting an entirely different object recognition library that may already exist in a more thorough state. In addition to identifying a greater number of background objects, the detector should also be trained with a much larger vocabulary. Instead of identifying an object only as the top-level classification, e.g. car, it could give a more specific identification, e.g. Honda Civic SI. This greater level of granular detail will not only lend itself to building longer, stronger passphrases, but it will also enable the program to output more diverse passphrases altogether.

VII. Conclusion

In this paper we aimed to define and analyze a new framework for password policy. Our policy attempts to address two competing aspects of password generation: password strength, addressed by an object detection implementation of the Correct

Battery Horse Staple method, and password memorability, addressed by providing users with the source image to serve as a reminder for future logins. Despite a somewhat incorrect hypothesis, preliminary findings are encouraging, particularly for the source image's benefits to long-term passphrase memorability. Subjects reported that not only does the image help memory in later logins, but its use in early logins serves as a study medium to reduce the users' reliance on it altogether. User memory results aside, the actual password generator aspect of the policy requires further development. To start, a functional integration of the *google_image_download* library is necessary to make our program a fully autonomous option for password generation. Further, a more complex model for object detection is required to output truly strong and unique passphrases. That said, it is evident that this process has the potential to be a superior successor to existing policies, particularly those still reliant solely on text based passwords. Based on these findings, we recommend future studies adopt a model with greater training and an expanded vocabulary of available words while maintaining the "image as a hint" philosophy.

References

[1] S. Williams, "Average person has 100 passwords - study," *SecurityBrief New Zealand*, 20-Oct-2020. [Online]. Available: https://securitybrief.co.nz/story/average-person-has-100-passwords-study. [Accessed: 6-Apr-2021].

[2] B. Meyer, "After analyzing 15 billion passwords, these are the most common phrases people use," *CyberNews*, 09-Apr-2021. [Online]. Available: https://cybernews.com/best-password-managers/most-common-passwords/. [Accessed: 6-Apr-2021].

[3] "Correct Horse Battery Staple Review - Password Advice - Virtual CISO," *Fractional CISO - Virtual CISO*, 01-May-2020. [Online]. Available: https://fractionalciso.com/correct-horse-battery-staple-review/. [Accessed: 7-Apr-2021].

[4] N. Woods, M. Siponen, "Too many passwords? How understanding our memory can increase password memorability," *International Journal of Human-Computer Studies*, Volume 111, 2018, Pages 36-48, ISSN 1071-5819, https://doi.org/10.1016/j.ijhcs.2017.11.002. [Accessed March 1, 2021].

[5] Y. Guo, Z. Zhang, "Optiwords: A new password policy for creating memorable and strong passwords," *Computers & Security*, Volume 85, 2019, Pages 423-435, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2019.05.015. [Accessed March 2, 2021].

[6] M. Siponen, P. Puhakainen, A. Vance, "Can individuals' neutralization techniques be overcome? A field experiment on password policy," *Computers & Security*, Volume 88, 2020, 101617, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2019.101617. [Accessed Feb. 28, 2021].

[7] J. Ellis, "Jacob Ellis," SandStorm IT, 14-Oct-2019. [Online]. Available: https://sandstormit.com/what-makes-a-strong-password/. [Accessed: 28-Mar-2021].

[9] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin. 1999. "The Design and Analysis of Graphical Passwords," *In Proceedings of the 8th conference on USENIX Security Symposium - Volume 8* (SSYM'99). USENIX Association, USA, 1. [Accessed March 19, 2021]

[10] G. Stenberg, "Conceptual and perceptual factors in the picture superiority effect," *European Journal of Cognitive Psychology*, vol. 18, no. 6, pp. 813–847, Feb. 2007. [Accessed April 1, 2021].

[11] C. L. Grady, A. R. McIntosh, M. N. Rajah, and F. I. Craik, "Neural correlates of the episodic encoding of pictures and words," *Proceedings of the National Academy of Sciences*, vol. 95, no. 5, pp. 2703–2708, 1998. [Accessed April 1, 2021].

[12] S. Chiasson, P. C. van Oorschot, and R. Biddle, "Graphical Password Authentication Using Cued Click Points," *Computer Security – ESORICS 2007*, pp. 359–374, Sep. 2007. [Accessed March 20, 2021].

[13] W. Meng, "RouteMap: A Route and Map Based Graphical Password Scheme for Better Multiple Password Memory," *Network and System Security*, pp. 147–161, Nov. 2015. [Accessed March 20, 2021].

[14] M. Olafenwa, J. Olafenwa, and DeepQuest AI team, *ImageAI*. GitHub/OlafenwaMoses, 24-May-2018. [Accessed March-April 2021].

[15] *How Secure Is My Password?* [Online]. Available: https://howsecureismypassword.net/. [Accessed: April 14, 2021].

[16] “Password Strength Test,” *My1Login*. [Online]. Available: https://www.my1login.com/resources/password-strength-test/. [Accessed: April 14, 2021].

[17] “Estimating Password Cracking Times,” *Better Buys*. [Online]. Available: https://www.betterbuys.com/estimating-password-cracking-times/. [Accessed: April 14, 2021].