CSCI 49381
Final Project Report
Fall 2022

Xing Chen, Dian Chen

December 2022

Abstract

Passwords are the lock and key to your house, a strong lock prevents bad actors from accessing sensitive information belonging to an user. Due to the inherent sensitive nature of passwords little is known about the characteristics of login request submitted to login systems. This make its difficult to make insights into user submission of passwords such as how often users submit incorrect passwords and how many users are utilizing passwords revealed in public breaches. Gossamer is a tool created solely to securely observe login requests with many considerations into the safety of the observed users. Previous observations on login systems with Gossamer deployed have indicated potential in the tool's use for designing and improving security mechanisms and policies.Our intent with this project is using the Gossamer paper and tool to replicate experiments and the gathering of predetermined data to make observations and understand the results. We will gather login information using a login system we created on a server hosted by Heroku in order to record measurements as done by Bohuk et al [1] but on a much smaller scale. To achieve our goal we recruited friends and workers from Amazon MTurk. Due to the nature of the experiment and we expect many inorganic results and because of this, prior to the data collection we made several predictions on the data we would receive.

1 Introduction

Gossamer is a tool created in order to safely monitor login services and generate data that would be not otherwise possible without compromising user safety. Our original intent was to fully implement the Gossamer tool using services provided by Amazon Web Services but we ran into several issues regarding this. Due to this we decided to pivot our project in another way to still try to replicate their experiment. Amazon MTurk is a place where real people can be hired to do simple task. Previous studies have employed the use of MTurk but there is clear indication that passwords recieved would not necessarily match normal password usages [2]–[4]. While this is no substitute for real logins there was no way for us to access actual login logs from a system containing sensitive information.

The authors of the paper focused very heavily on security and safety of the users, they implemented many features because of the fact that there were sensitive information to protect. Their study of passwords and login requests were conducted at two different universities and involved the modifying of Gossamer to fit the security needs of the universities in question. The data gathered came from actual students with real sensitive information behind their login and passwords. To prevent the data from being used maliciously if it happened to fall into the hands of bad actors was a primary concern, there were many layers of security such as an ephemeral MYSQL database and periodically erased all stored data. They even had to scrap a previous for logging data as it would improve the efficiency of brute force attacks by a large margin if it was included. In the paper they use Gossamer logs to analyze data we will try to replicate this to a degree.

Our replication of their experiment will not involve any sensitive information held within user accounts and thus security was not our paramount concern, however, with a new approach we did put some thoughts into making the data we are gathering such as combination of usernames and passwords less sensitive. The prerequisite items for our approach to password analyzation are as follows, a login system consisting of a login page, register page, successful login page, MYSQL database for username and password storage.

2 Approach

The first step of our experiment was to create an appropriate login system so we can begin tracking and logging accounts that log in, this was done before we even attempted to implement Gossamer. In our proposal we were planning on creating this in PHP but we settle on Python instead because we were more familiar with the language. At first as evident by the existence of the email field we were collecting a username, password, and email. Due to the fact that emails can somewhat be considered sensitive information as they are personally identifiable information (PII) and do provide some utility to bad actors we decided to remove emails from our database. We did however leave the email field as a requirement when creating an account to give the impression that the created account will be tied back to the user so it would encourage better password etiquette. In hindsight we should have known not to collect email information as they were already deemed to be PII in the original paper. We did not provide any strict guidelines on proper username and passwords for user accounts to not provide a great deal of influence in their selection.

As we were gathering data we looked over the usernames that were being used we deemed it appropriate to keep, our reasoning for such actions despite what the researchers did was the usefulness of the data. In the original study each username was tied to an individual student but in regards to our data we determined that it was not PII. We did have access to IPs of users but we did not investigate or log that information because we deemed it personally identifiable.

As we did not provide any guidelines for password strength to users, we predicted passwords used to be relatively simple and weak. We also thought that there would be a large number of "randomly" generated passwords automatically provided by browsers like Google Chrome. We compared the gathered passwords to top 5000 most commonly used passwords in Rockyou and Hashcat leaks [5], [6], Bohuk et al provides another list of 1000 commonly used passwords.We had access to encrypted passwords but our reasoning behind this is the data we will be collecting using Gossamer has no importance or value since the purpose of these login submissions is simply to learn about password habits.

We wanted to try out certain methods in breaking our system such as credential stuffing as mentioned by the researchers.

3 Attempt

Within our implementation of the Gossamer, we first started with our log-in server, and through research and some effort, we were able to set up a functioning one. Once we came to the implementation of Gossamer as an application into our log-in server, we face a variety of issues. The largest of the issue is that implementing the actual application, as the Gossamer that is found on GitHub was adjusted to the deployment on the two university where it was previously implemented in, thus, we needed to adjust it greatly if we wanted to implement it within our log-in server, leading to the issue at hand, where due to our limited knowledge on how to properly adjust the code from Gossamer in order to implement it. This seemingly resulted us in having two choices to make, either to continue with the current log-in server and research Gossamer further in order to attempt to make it work, or to attempt to build another log-in server using a different system to hope for a different, but compatible, outcome.

As the time allotted for the project drew near, we realized that both options weren't viable, but rather, it would most likely not work in both cases. Instead of our original proposal, we decided to pivot slightly. We still wanted to work on and further take a look at the previous implementations of Gossamer and what implications it could have for the future of digital security. Hence, instead of getting our own sample, we wanted to go in-depth on an analysis of the previous implementations of Gossamer, along with a reflection on where our process went wrong and where we could go in the future learning from these mistakes. Thus the rest of this report covers what specifically went wrong coupled with more details on either a logical reason or based on a certain experience that our implementation would have given inaccuracies in the first place. Along with the reflection, we went on further to analyze the report given by the previous implementations of Gossamer, along with a further look into the graphs and results provided, so that we can also draw our own conclusions on the future implications Gossamer truly holds in either helping or harming the digital security field.

4 Challenges

(Implementation + Time) Given a much shorter time frame compared to that of the creators of Gossamer, we knew we were unable to perform to their scale, such as collecting huge sample sizes of data. Unfortunately, the time constraint also limited our knowledge in the implementation of Gossamer. Login servers need to usually be secure web infrastructures to exist, so that the website that we base off it can run properly. Our first issue came in the form of acquiring the knowledge in building a proper login server within the given time constraint. In the acquisition of the knowledge, most of our time then went into designing the server. Then the next issue followed: the implementation of Gossamer. The Gossamer application requires heavy adjustments to its original code in order to have it be compatible with our login server, or the other way we could make it would be to design the login server in order to be compatible with the original code of the Gossamer application with little changes. Given that we had created our login server, we attempted to implement Gossamer. Unfortunately, we were unable to implement the Gossamer application within our login server due to the compatibility issue. With this issue at hand, we were unable to proceed until either we redesign our login server or to edit Gossamer in order to fit our login server, which would require another heavy time commitment that we don't possess. Given the time constraints, though we were able to understand the basic fundamentals of Gossamer and how the application is used along with its result and to be able to construct a functioning local login server, we could not effectively execute the initially intended project.

(Human Bias) Based on the initial plan to recruit users to create login from Amazon MTurk, a flaw in the experimentation soon came to light. Within a crucial digital environment, such as that of the original Gossamer implementation where it was attached to that of University login, there often exists an incentive for individuals to create a secure and safe password. Whether the intent is to not let others view personal information, a matter of pride based on academic or economic ability, or to secure academic or financial decisions to only the user, the incentive is often strongly tied to how securely the user creates their password. Given this correlation between a strong incentive and strong or weak passwords, building a blank website, whose only purpose is to ask users from Amazon MTurk to create accounts and login with no other incentive, we realize that we will often be left with extremely weak passwords, since there offers no incentive to not have their account be breached. This influx of weak passwords would further skew the result given by the implementation of Gossamer, ultimately giving us a huge inaccuracy for the comparison of between the original implementation. Thus the experiment, even if fully executed, would produce extremely skewed, and controlled results, due to the existence of human nature and our failure to replicate it within a controlled environment.

(Attacks + Other Real World Scenarios) Within Gossamer's implementation, they also experienced cyber-attacks, with public breaches that gave away information on either usernames, pass-

words, or both. During these attacks, as seen in Figure 2, there existed a high volume of failures. This is due to the attacks often changing the password on the account, thus "stealing" away the account from the user, whether it is temporary or permanent, depending on the website developers. In a short-term controlled replication, there exists an extremely low chance that real-world factors, such as attacks and breaches, can occur. In turn, the success/failure rates will be skewed, limiting the potential of failures to user input and memory of their password and account. In essence, due to the controlled environment of the implementation, there exists a lack of real-world factors that are left unaccounted for, thus causing an inaccuracy in the data, most likely in favor of the success rate. With this, we cannot confidently say that the data produced would be reflective of the general populace, nor can we accurately compare it to the original implementation of Gossamer.

(Number of Results) The number of useful data points will be a limiting factor for our analysis, we are simply unable to achieve a large enough dataset with the current methods compared to what Bohuk et al did. This is further exacerbated by the fact that it is very likely that a large portion of our dataset will fall into simple and non organic passwords. After separating the dataset into groups we could be left with very little of what we deem to be organic passwords to work with. In hindsight it would have been better for us to implement a password strength requirement, however at the time we thought that even those easy passwords would be considered useful data. Repeat users also fall into this section of challenges, it is very unlikely that we get users who log in multiple times, if anything we believe that it repeating logins from MTurk if there were any, may have a high chance of just making another account instead. Although this metric is no longer relevant because we failed to implement Gossamer and will not be using user IPs.

(Safely Collecting Data without Gossamer) Since we were not able to implement gossamer we had to dial back on some of our expected avenues of data collection. Firstly we had to determine what sort of data we can even gather in the first place without compromising the privacy of the participating users. The list of items we could then interact with was reduced heavily despite this, we still wanted to mimic some aspects of Gossamer in our own methods and collect data while still protecting the users. Originally we had intended to gather much more in depth data regarding individual users but now that is no longer safe to do without having access to many pieces of PII which we are reluctant to work with. Information that we can no longer gather include unique users because of IP/Email instead we had to substitute that data point with unique accounts.

5 Security Analysis

5.1 Gossamer

Due to the high sensitivity of user passwords especially if it is connected to access of sensitive information, it is often quite difficult to analyze what is submitted to a login system in a real-world environment. The way Gossamer avoids any dangerous actions that may expose sensitive information is by forwarding the information sent in a given login submission to a temporary database, where it is then further processed. The type of information being pulled from a login request and sent to this special database includes the user's inputs for username and password, as well as an IP address, login results (whether successful or failed) and other parts of information pertaining to the login. A measurement service is then used to process all this information in a secure manner, where what is produced will then be sent to a persistent database. The measurement service accomplishes this by storing all processed data on sanitized logs so that no confidential information is present on the persistent database. Only those conducting the study should have access to the database containing the clean, anonymized login submissions. This architecture is visually represented on Figure 1 below, courtesy of Bohuk et al. It is important to note that although the logs produced by the measurement service in use by the Gossamer tool sanitizes any sensitive information it may contain, the content of these logs may make re-identification attacks possible [1].

There are other security considerations that the authors have made, given the sensitivity of their

research. The authors explicitly state the four main security properties that their tool must conform to, as a means of resisting a variety of attacks. These principles are least privilege, bounded-leakage logging, periodic deletion and safe-on reboot. The principle of least privilege is one of four principles put in place in order to ensure that those using Gossamer only have access to necessary information pertaining to login behavior. With bounded-leakage logging, the system logs of statistics pertaining to user passwords is carefully designed to limit the benefits it may provide in guessing attacks if data from the analysis service were to be completely compromised. Additionally, with periodic deletion any data stored longer than 24 hours is removed from the system. Rebooting also results in the erasure of all sensitive data stored from the tool, which is made possible due to where the ephemeral database is stored - a memory-based file system. This satisfies the safe-on reboot principle stated by the authors.

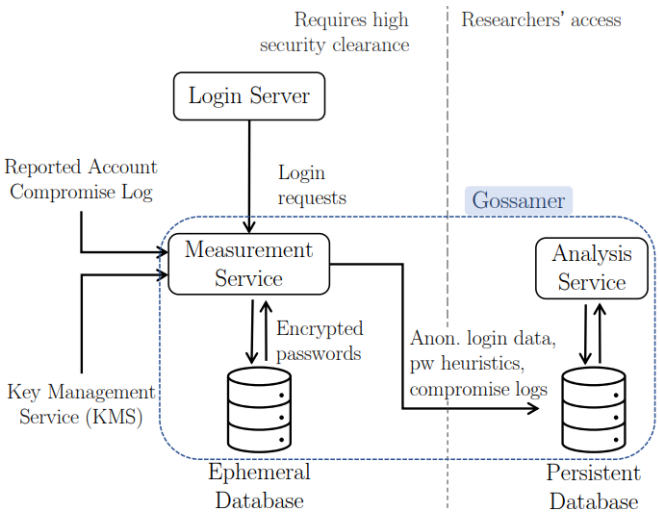

Figure 1: Components of Gossamer

5.2 Our Attempt

We created the login system with the intention of adding Gossamer on top, we also wanted to collect information and work with them through Gossamer logs where it becomes much more secure. However because of our failure to implement Gossamer on our systems we had to determine safe ways for us to study the end result without having access to PII of users. We made a mistake because we stored all usernames, passwords, and emails were stored in plain text. While we did remove emails, we did not move off the use of plain text.

In an attempt to not risk the safety and to follow some fo the safety guidelines set by Bohuk et al we limited ourselves to the access of certain data. There was a plan to use user IPs to determine the number of unique visitors compared to the number of accounts created but idea was tossed because IPs are considered PII and we probably should not have access to them or log them. We ended the collection of emails a little bit after a few days because the that risk had slipped our minds despite removing that collection of emails but leaving the field intact on the register forms. We thought we should scramble passwords and usernames so that they do not give us any actual information but upon consideration we thought there was little risk.

6 Predicted Results

Prior to starting our data collection we made some predictions on what kind of results we expected to receive from users. We wanted to see if we could accurately predict what other users would do based on what we would do. We also expected that data from MTurk would differ drastically from data from friends we ask to log in for us, we are hoping to be able to recognize them. At this point we still had the intention of collecting emails which we later decided was a bad idea. We also did not know how many results we were going to end up with.

For emails we predicted that the entries we would receive would be very simple and very likely emails that do not belong to the user. This is prevented on actual website logins with a email verification but since we do not have that feature users may test account creation with a non real email. We expected many results from automatic email generators or very commonly used simple emails (i.e abc123@gmail.com). We predict that while there may be some valid emails belonging to the user in the data, a majority of the results would be fake. These predictions are somewhat inline with what we would do on unknown sites asking for emails on account creation in order to prevent spam. Ultimately this did not matter in the end because we decided to scrap the idea of collecting email addresses because valid user emails are PII, we had forgotten to consider this fact when creating our form, but the idea was quickly brought up when it went online. We decided to discard the email field in its entirety and all previous emails we had already collected were deleted for user safety. Despite us no longer collecting emails we decided to not remove the email field on account creation, we thought this would give the idea that accounts could be tied back to the user and provide us with more data that we consider inline with actual accounts.

For usernames we predicted since that the database only had our test logins and basically every username would be available the data we would get from usernames would be all over the place. We could expect many usernames to be very simple, something along the lines of (name1,name2,abc123) or maybe resembling the password they would decide to use. There will of course be people who have set online usernames that they already use for every site, while we are unable to fully determine whether or not an username fits that criteria, we will attempt to look out for them as they are probably the data points that will prove to be the most useful. We also expected to see a clear difference in username etiquette among our friends compared to people logging in from Amazon MTurk, this may be due to the fact that we had to explain part of the project to our friends before they were given access to the login system. This may result in them being more inclined to use proper names or simply their names as the login username.

Regarding passwords we had expectations of three distinct groups of passwords that would be created by the user. Group 1 would be classified as very simple passwords. Examples of these passwords are the ones available on the most common password lists provided by Hashcat and Rockyou [3], [6] and a separate list provided by Bohuk et al. Another group of data that will fall into group 1 will be passwords that are the same or very similar to the username. We expected many of our results to be on these list as there is simply not an incentive for users to come up with stronger passwords. Group 2 passwords would be passwords created by Google password manager or other similar services. These results will look like a string of absolutely randomized ascii characters and will be very strong in terms of security. They have the distinct characteristic of looking very inhuman and quite impossible for a regular user to recall, they should be easy to classify and group. Group 3 passwords would be what we deem as actual user created passwords, this may be a bit subjective so we decided if there werent enough results in group 3 it would extend to anything that was neither in group 1 or 2.

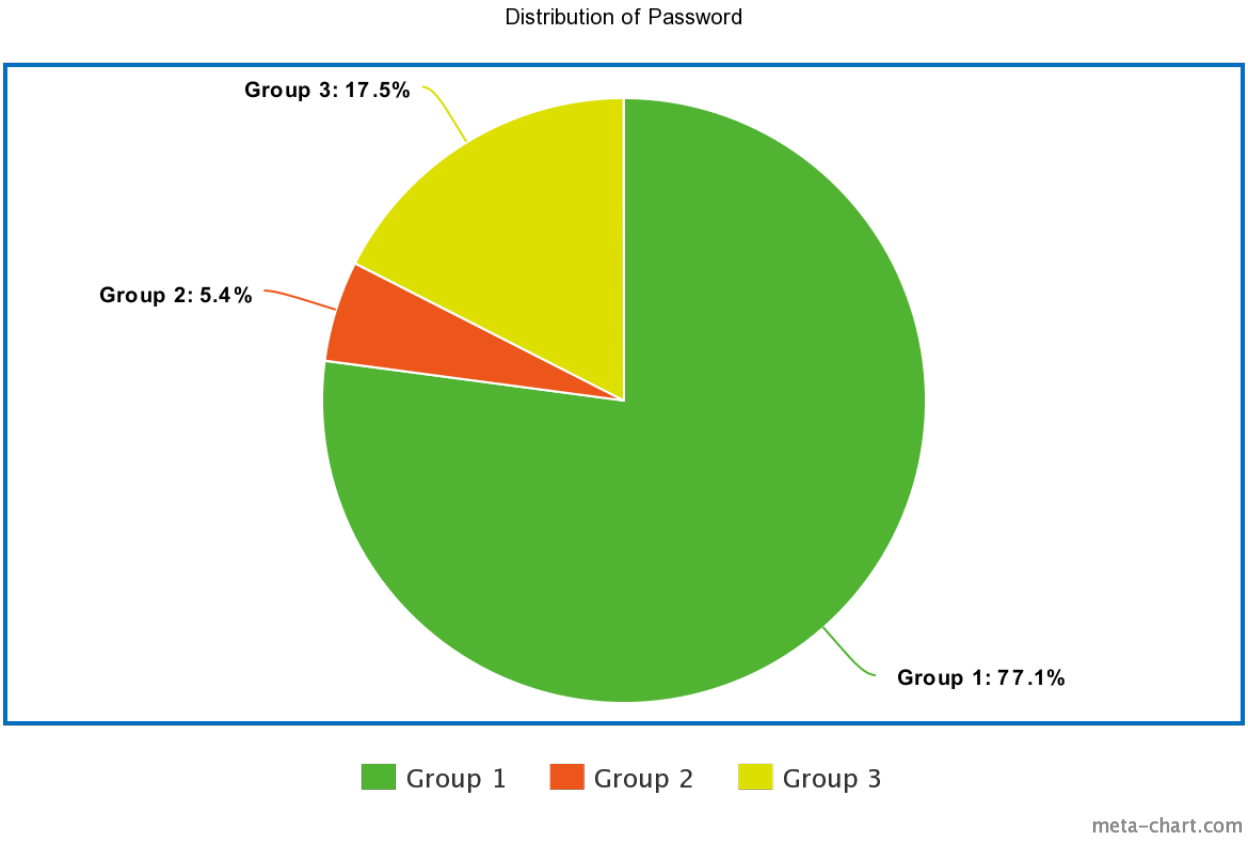

Figure 2: Distribution of Password Groups

7 Data Derived Measurements

7.1 Preliminary Anaylsis

Upon preliminary analysis can immediate see a few pieces of data, we received a total of 612 responses of unique usernames, this number is a bit lower than our visit count according to Heroku. We could also immediately see that some of the usernames we recieved were also exactly the same as their respective passwords. There were also users who decided to use an email address as their username even though that was not specified and there was an email field. For cases like that we decided to remove everything after the "@" symbol so we could still use that data point. The shortest password was simply a blank space character. Upon a brief review of the data we believe that would have been a wise idea to implement some sort of password strength requirement when creating an account as such simple results are simply meaningless to our data. The longest password was 20 characters and made up of random ascii characters, we believe that this was created via the use of a password manager.

There were multiple instances of usernames that are a simple word followed by a number with other usernames that were the exact same save for the number at the end. There is an instance where it followed that format of username the password was exactly the same between two instances. We believe it is safe to assume that they were a repeat user that might not bothered to login to their old account and just made another. Originally we thought that they just forgot their login info and we do not provide password recovery services because we weren't actually collecting their emails,

but that does not add up because the username and passwords are otherwise the same. There is an instance of this username followed by a number but password was created by a password manager, this is likely another repeat user but we cannot fully determine if this is true, it may simply be an overlap in username. Overall there were not many what we deem to be password manager generated passwords. Our prediction of majority of passwords being simple and on the list of leaked passwords were correct.

7.2 Analysis of Measured Data

After analyzing the data we had collected we ended with 472 passwords (Group1) that were available in the 3 list we were provided, 33 (Group2) passwords are believed to have come from password managers and the other 107 (Group3) passwords we believe to be resembling actual passwords. Respectively they are 77.1%, 5.4%, and 17.5% of the collected data. This result is actually better than we expected, we expected more users to use simple and insecure passwords because there is no incentive.

For our predictions regarding username they were for the most part accurate. There was a large set of usernames that were not given any care or thought. Its very likely they had no intention of coming back after the task was completed. We did see some usernames resembling actual usernames but it is hard to determine whether that is true.

This distribution of passwords shows that if someone had access to a large list of usernames credential stuffing attacks and other brute force methods would be incredibly effective on our systems. Due to the lack of information we hold on users combined with the fact that majority of usernames were non unique there is likely little risk.

8 Future Implications and Implemenations

Given a future opportunity to work with this technology, we can possibly implement a password policy suggestion. This policy suggestion will prompt user to use a stronger password, pushing for them to meet a certain or predetermined recommended checklist for a strong password. In addition to that system, we can have Gossamer cross check the users' created password with hashcat, rockyou, breach password to user created password, as seen in the image attached below. If matched, it will stop the user from creating the password completely and prompt them to change the password and elaborate on that the password is either extremely weak or has been found within a recent breach or attack, resulting in a higher susceptibility that the password and account can be breached. Within this system, there is an aim in order to eliminate weak passwords within the websites, and hopefully encourage the usage and memory of stronger passwords.

Ultimately, we can also try to push for the use of password managers which creates strong passwords and then is stored for the user so that the user will not need to constantly recall this particular strong password. We can also see and monitor the password manager users with users who have strong passwords along with a low failure rate.

Once given more time and resources, many of the issues mentioned within the Challenge Section can be resolved as the following:

Given more time to research and further understand how to utilize Gossamer and how to ultimately build a login in server which is compatible with it, Gossamer can easily be implemented in to start collecting data and to result in a report from the system. To do this, a time commitment is a must, along with asking the creators and those who worked on the original implementation of Gossamer to assist in this local implementation that we have attempted. Given also a further allotted time, Gossamer can be ran on the website for the same time frame as the original implementation for a more accurate comparison, or for a longer time frame so that there exists a larger sample size, along with witnessing any consistent trends or anomalies that may appear given a longer timeframe

than that of the original experiment. In this scenario, that Gossamer can run for a longer time, we can also witness any other changes or events in the real world that could possibly affect a user's valid/invalid input of their password in an ever-digitalizing world.

As discussed previously on a further allotted time to understanding Gossamer as an application, we could also use that time to either partner with a company or to build a website that will offer some sort of incentive in order to combat this existence of human nature. Some sort of incentive, such as that of an exclusive coupon or a code to enter in order to win a giveaway of some sort for a gift card and such, can easily incentivize humans in order to procure a safe and secure password, resulting in a more accurate result in order to compare, as now the subjects will have an thing to want to protect and securely create a password for. This will result in a more reflective result report given by Gossamer.

Whilst there are tons of real-world factors that may affect a password's strength, along with the success and failure rates of user login, attacks and public security breaches are on the top of that list, in both their frequency along with the amount it can affect the typical trend of success and failure rates. Due to the controlled nature of the experiment, the website will most likely not receive any attacks, thus we can artificially create attacks and breaches. Within this artificial attack, we can change the users' passwords, then take note of the rise in failure rates, along with having the option on our developed web-page in the process of changing their password back. This can directly affect the process along with the resultant actions taken in order to get the account back for the user. The rise in failure rates should also reflect that of in a real attack. Due to also the nature of this experiment, the subjects won't be informed about any artificial attacks prior to them happening, as that may skew the results as well as who decides to participate.

(Password Strength Requirement) From the data we gathered it became pretty obvious that we should have implemented some sort of password strength requirement. Even a basic lower character limit would have helped improve our data. Even though the characters like " " are listed as a common password on the lists provided it was not a valuable piece of evidence. It would not have affected the majority of common passwords. Even that simple barrier may have promoted users to be more serious with their passwords and increased the percentage of group 3 passwords we recieved.

9 Team and Division of Labor

Our team is comprised of two members. Certain task were redistributed during the process of the experiment.

Xing Chen
Tasked with creating login page.
Data Analysis
Project Report

Dian Chen
Tasked with hosting login system.
Presentation, Poster
Project Report

10 Conclusion

The utilization of Gossamer as a security tool is an interesting topic to build upon. We did not believe that we could build or collection login system and submissions on the same level as Bohuk et al, however, the conclusion of our project ended up a larger failure than what we could have predicted. We were not able to correctly implement the tool onto our login system and instead

built a greatly simplified version of our original proposal. We are disappointed in the outcome of our project. We had hope to do more in depth analysis of incoming data especially relating to the login process and user habits. Despite the failure we did learn a number of things from our findings, the reason every respectable website has such strict password requirements. Users will always gravitate towards the easiest and most simple passwords simply because they do not care. Password strength requirements are very vital as a first line of defense especially brute force attacks. Password managers are incredibly useful and provide very strong passwords without any effort on the user end. There may be complications that include losing access to those stored passwords due to various reasons but generally it would be wise to employ the use of one. Users will simply provide information if you ask nicely. They may even provide more information containing sensitive data entirely willingly without prompt.

References

[1] M. S. Bohuk, M. Islam, S. Ahmad, M. Swift, T. Ristenpart, and R. Chatterjee, "Gossamer: Securely measuring password-based logins," in *31th USENIX Security Symposium (USENIX Security '22)*, USENIX Association, Aug. 2022. [Online]. Available: https://www.cs.cornell.edu/~marina/Gossamer.pdf.

[2] B. Ur, P. G. Kelley, S. Komanduri, *et al.*, "How does your password measure up? the effect of strength meters on password creation," in *21st USENIX Security Symposium (USENIX Security 12)*, Bellevue, WA: USENIX Association, Aug. 2012, pp. 65–80. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/ur.

[3] S. Komanduri, R. Shay, P. G. Kelley, *et al.*, "Of passwords and people: Measuring the effect of password-composition policies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11, Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 2595–2604, ISBN: 9781450302289. DOI: 10.1145/1978942.1979321. [Online]. Available: https://doi.org/10.1145/1978942.1979321.

[4] P. G. Kelley, S. Komanduri, M. L. Mazurek, *et al.*, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *2012 IEEE Symposium on Security and Privacy*, IEEE, Oct. 2012, pp. 523–537. [Online]. Available: https://ieeexplore.ieee.org/document/6234434.

[5] N. Cubrilovic. "Rockyou hack: From bad to worse." (Dec. 2009), [Online]. Available: https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/.

[6] J. Steube and G. Gristina. "Hashcat." (), [Online]. Available: https://hashcat.net/hashcat/.