Algorithm for User-friendly Password Policy

Jorge Hernández Liang
Master in Computer Science
Dr. Alfredo Cruz
Electrical & Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— Password based security has become prevalent as the de facto measure against intrusion. Computer systems rarely focus on usability, and little research has been done to change the current, near universal method of implementation. This paper explores the weaknesses found in password based systems and how they can be mitigated. This will conclude with a new set of algorithms for validating passwords and creating baselines for setting minimum required strengths for password defense. This new metric allows for a wider diversity of possible passwords while maintaining security by assigning value to certain characteristics and requiring a minimum amount of security be achieved before accepting the password instead of relying on static requirements in minimum length and presence of character types. The new approach embraces password diversity and allows for trade-offs of different password elements in order to allow users to use a wider array of strategies at the moment of generating their passwords.*

Key Terms *— Computer Security, Passwords, Privacy, Usability.*

INTRODUCTION

Passwords are a scheme in which users are granted authorization to access specific information or systems by providing a specific string of characters. They have become ubiquitous in our lives, but are also frequently exploited for common weaknesses. This paper explores a new approach to password security by presenting a new scheme for password requirements. This new system adopts a variable password strength minimum based on expected threats as well as multiple ways to reach those strength requirements. By not forcing the user to implement specific rules within their passwords, the new scheme hopes to undo certain forms of conditioning present in many current passwords.

The new system borrows some strategies found in previous NIST publications [1] in using entropy as a metric for the value of characters, but changing the way the value of characters and bonuses is calculated, as well as adding a way to calculate minimum levels of uncertainty required to accept a password. This allows the user to negotiate the contents of their password, including length and diversity in character categories. The new scheme also allows users to generate passwords that far surpass requirements to allow for longer duration of use, achieved by requiring complexity gains beyond the additionally granted lifetime.

COMMON PASSWORD CRACKING WEAKNESSES

In order to understand what improvements need to be performed when creating our new password scheme, it was important to understand the common vulnerabilities that are exploited in password attacks? The primary issues that are exploited are length and predictability of characters, both of which are easily addressed by changes to password requirements.

The clear example of where length is an issue with passwords is observable in **bruteforce password guessing attacks**. These attacks exhaustively attempt to verify the password keyspace (every possible combination of characters that can legally form a password). Due to the exponential growth nature of password space, increasing either length or characterspace (all possible characters usable in the password) even small increases will have a significant effect on the complexity of the password. This can be seen in Table 1, where we display a chart of keyspaces based on what characters are present and the length of the password. This table shows the rapidly growing nature of passwords. The left column, representing the base number of the exponential growth, has a significant impact as it grows. Even the smallest growths in the amount of available characters has an impact of orders of magnitude when looking at the rate of growth for password keyspaces. The middle block of the column builds off of the left characterset size column by explaining how each size is constructed, using the different character types as groups (Digits in decimal or hexadecimal form, singlecase or mixed case letters, standard keyboard special characters, and ASCII characters) which are then summed up to provide the characterset size. Finally, the rightmost block serves the primary purpose of the table by displaying the password keyspace created by those different charactersets using different password lengths. These are all presented in scientific notation as the order of magnitude proves to be critical component, since many of these charactersets grow very quickly. It is also important to note that their rate of growth increases faster on the lower rows due to having a much larger characterset, and therefore a larger base number for the exponential growth.

Predictability proves to be an issue for password systems due to it reducing the randomness that is inherently critical to password based security. It is harder to address, but we can attempt to understand how it came to be and avoid repeating those mistakes. In a previous study [2], it was learned that the characteristics of user passwords is reflective of common password requirements and manifest in similar ways across many users. The same study also presents evidence that user diversity could lead to password diversity when freedom is allowed at the time of password generation.

Passwords are often presented as requiring at least 8 characters, both upper and lower-case characters, and the presence of either numbers or non-alphanumeric characters. This is shown by passwords consisting of at least 8 characters in most users and frequently meeting the requirements through the presence of an uppercase character in the first character of a password and the insertion of a special character at the end of the string of text. For example, replacing “password” with “Password1”. This conditioning can be referred to as “Pavlovian Passwords” due to similarities to classical conditioning [3].

Table 1
NIST Keyspace Cardinality Calculations
Reprinted courtesy of the National Institute of Standards and Technology, U.S. Department of Commerce.
Not copyrightable in the United States.

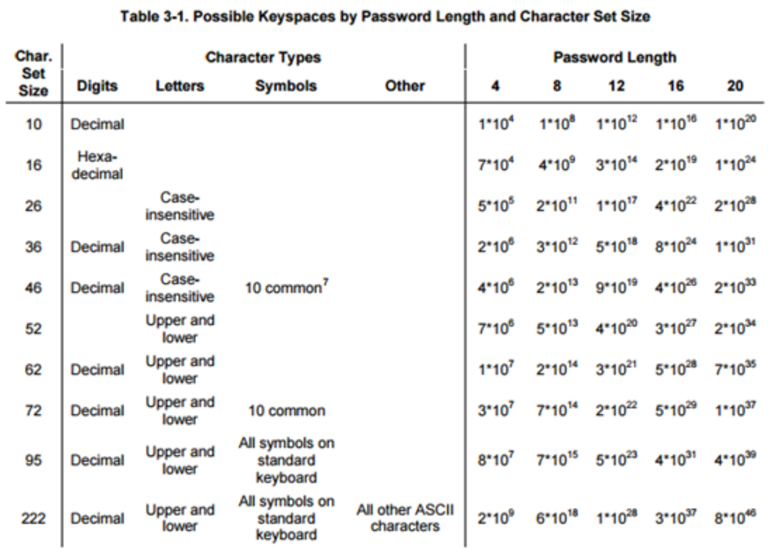


Table 3-1. Possible Keyspaces by Password Length and Character Set Size

Char. Set Size	Character Types				Password Length				
	Digits	Letters	Symbols	Other	4	8	12	16	20
10	Decimal				$1*10^{4}$	$1*10^{8}$	$1*10^{12}$	$1*10^{16}$	$1*10^{20}$
16	Hexa-decimal				$7*10^{4}$	$4*10^{9}$	$3*10^{14}$	$2*10^{19}$	$1*10^{24}$
26		Case-insensitive			$5*10^{5}$	$2*10^{11}$	$1*10^{17}$	$4*10^{22}$	$2*10^{28}$
36	Decimal	Case-insensitive			$2*10^{6}$	$3*10^{12}$	$5*10^{18}$	$8*10^{24}$	$1*10^{31}$
46	Decimal	Case-insensitive	10 common[7]		$4*10^{6}$	$2*10^{13}$	$9*10^{19}$	$4*10^{26}$	$2*10^{33}$
52		Upper and lower			$7*10^{6}$	$5*10^{13}$	$4*10^{20}$	$3*10^{27}$	$2*10^{34}$
62	Decimal	Upper and lower			$1*10^{7}$	$2*10^{14}$	$3*10^{21}$	$5*10^{28}$	$7*10^{35}$
72	Decimal	Upper and lower	10 common		$3*10^{7}$	$7*10^{14}$	$2*10^{22}$	$5*10^{29}$	$1*10^{37}$
95	Decimal	Upper and lower	All symbols on standard keyboard		$8*10^{7}$	$7*10^{15}$	$5*10^{23}$	$4*10^{31}$	$4*10^{39}$
222	Decimal	Upper and lower	All symbols on standard keyboard	All other ASCII characters	$2*10^{9}$	$6*10^{18}$	$1*10^{28}$	$3*10^{37}$	$8*10^{46}$

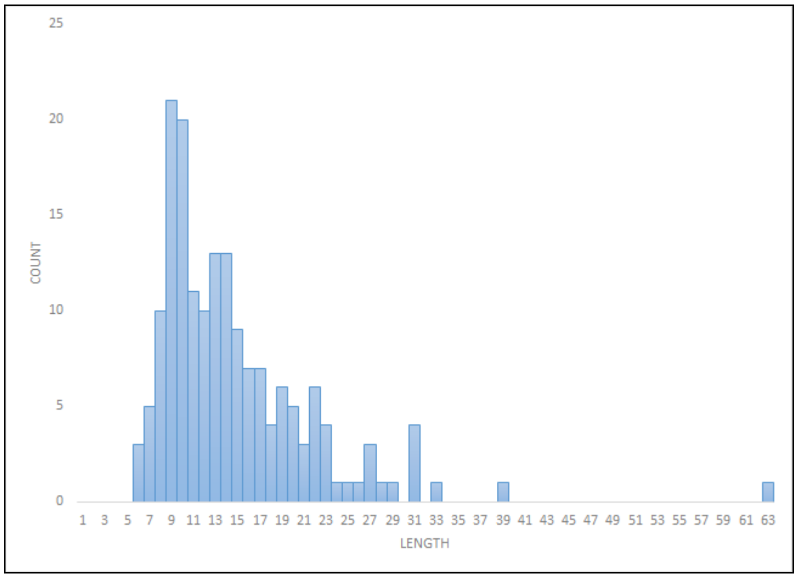


Figure 1
Submitted Password Lengths

From here, strict password rules have led to the issue of predictability and therefore we must avoid repeating the mistake in future attempts to craft password rules.

PREVIOUS WORK

As mentioned, the author had performed a related previous study on password based security [2] that demonstrates some useful information for this topic. The study consisted of an experiment were users provided sample passwords through a survey that was answered by university students. The survey consisted of requests for non-personally identifying demographic information, such as age range, primary language, gender, and if they had a work/study background in technology. They were also asked to provide a mock password, but were not given any restrictions as to the contents of the password. (The results of the survey can be observed in Figure 1.) The passwords were checked for several characteristics. The passwords were then placed in to categories based on their relative strength levels using traditional criteria.

The users themselves were also grouped based on provided demographic information. This allowed for the verification of the level of influence that demographic information plays on the provided password characteristics.

Several conclusions were found such as gender influencing length of the password, employment and education background influencing the overall strength, and other factors such as primary language being less significant.

Another finding was that the average password length of the participants was higher than previously recorded averages found, which seems to be consistently increasing as time goes on. This effect might be amplified within this study as participants were relatively young, being composed of university students. The distribution of their password lengths can be seen in Figure 1. The significant points of information in this graph are the low frequency of short passwords. The shortest instance of password length was 6 characters long, and was not very common as a password length. It's mode of 9, with 10 characters being a close second also brings positive information, as these appear close to the leftmost edge of the values and are also very close to the average of password lengths. The average itself also gives information, as the length of over 14 characters on average continues the trend of increasing password lengths as time goes on [2].

PREVIOUS ATTEMPTS AT STANDARDS

There have been two significant attempts to standardize the realm of password generation, one by the Department of Defense [4] and again by the National Institute of Standards and Technology [1]. The first mentioned was the Department of Defense Password Management Guideline (also known as CSC-STD-002-85 or simply as “The Green Book”) [4]. This text contains many of the recommendations observed today for password management, such as the recommendation of passwords being replaceable by the user and that they should not be written down or re-used. The rule for minimum length of 8 may also have originated from one of the examples presented in the text and was simply never updated as technology grew. This highlights the issue of non-evolving standards for passwords, which is particularly critical in the constantly evolving environment of computers, were hardware is constantly changing and the speed to compute these passwords is quickly being reduced.

The NIST attempt came in their special publication 800-63-2: Electronic Authentication Guideline. Here they try to use as a base the concept of Shanon Entropy (a measurement for unpredictability) to create a variable scheme for password approval. According to this document’s strategy, characters would add a certain number of bits of entropy (unpredictability or randomness) based on their position, with bonus bits being awarded to the password based on fulfilling addition criteria such as having a mix of upper and lowercase characters. This scheme is conceptually solid, helping to solve some of the presented issues, but it does present some shortcomings. There is no presented way to calculate a minimum acceptable level of randomness for a password, the calculation for level of randomness based on position of characters is not currently supported by established research, and there is no incentive for users to create a password beyond the minimum required to be considered acceptable by the system.

PASSWORD RESILIENCE TESTS

In order to demonstrate how the passwords execution times grow quickly when the keyspace is altered and how quickly even minor changes have an impact on execution time, it becomes necessary to demonstrate the runtime with concrete examples.

A series of tests were run to calculate execution time of brute force attacks on a computer with hardware and software representative of what might be used by a common password cracker at the time of execution. The specifications for the computer were as follows:

- Intel Core i5 6600K CPU.
- MSI Z170A Motherboard.
- 16GB Dual Channel DDR4 RAM at 3,000 MHz speed.
- NVIDIA GeForce GTX 1070 GPU (999 MHz clock, 8192 MB GDDR5 memory, 1920 CUDA cores, Driver ver. 376.33).
- Windows 10.
- Hashcat V3.2 Password Cracker (MD5 Algorithm).

The tests were computed on passwords representative of real world scenarios such as Amazon password requirements (6 character, both upper and lowercase), Facebook requirements (6 characters), common password requirements such as minimum 8 characters (both any capitalization and mixed) as well as for other reasons such as being resilient to specific methods of attack and more. The primary distinction for character sets is seen in the name of each group. Categories identified as Basic# require only 1 letter case, Complex# requires both uppercase and lowercase, and a special character or number. This is in addition to length requirements, which is the number in the category name.

The results of these tests can be seen in Table 2. Where the estimated execution time for an exhaustive attack on those spaces is shown.

The result of these is showing that there are clear breakpoints where there are significant gains, enough to add a margin of safety beyond the minimum requirement.

Many of the simple requirements have short execution times on current hardware, whereas the more complex requirements presented a rapid increase in length.

The results show that both the inclusion of diverse character sets and length add significant amounts of security. Their combined effects produce a rapid growth that helps greatly with difficulty at guessing the chosen string. With simple character sets such as only lowercase characters, there is an acceptable strength, taking approximately 414 days to exhaustively search the password space in our tests, but increasing from 16 to 21 characters drastically improves the search time to approximately 1156. Similarly, when using a complex character set such as all uppercase, lowercase, numbers and non-alphanumeric leads to execution times of a few seconds at 8 characters in length, but provides similar security to the Basic21 results at approximately 1166 days for the exhaustive search.

Table 2
Runtime Estimates

Complexity	**Runtime (in days)**
Amazon	0
Facebook	0
Basic8	.0001
Complex8	1.3
Basic11	2.3
Basic16	414
Complex10	1166
Basic21	1156

Goals for Guidelines

The basis for password security is unknowability and unpredictability. By requiring information accessible only to legitimate users, it becomes impossible for attackers to gain access without coming across the password by chance. Restoring this state of unpredictability and minimizing the possibility of an attacker successfully finding the password is the primary goal of this scheme.

The first issue that can be addressed is length, setting a minimum length that prevents the practicality of a brute force attack or its variants allows the password to perform its most basic function. For this reason, the password must adhere to a minimum length even in a worst-case scenario where the character complexity is at a minimum.

Lowercase character only passwords, however, are not necessarily a given. Many users can implement passwords that have a wider range of characters and that should be rewarded as it provides significant additional security. For this reason, passwords with lesser length but more complexity should also be allowed. These tradeoffs must be offered to prevent the case of Pavlovian passwords mentioned previously. The added security can manifest as both mixed character capitalization and the presence of numbers and non-alphanumeric characters, all of which should be treated as separate cases. Although, it is important to always maintain that the increase in security should be comparable or greater than the loss in security from predictability. To avoid abuse in extreme cases, it becomes beneficial to treat these benefits with diminishing returns to maintain a minimum length expectation.

Users should also be encouraged to go beyond a minimum requirement expectation for safety, unfortunately previous attempts to force this have resulted in the mentioned predictability [2] and therefore an optional reward system can be embraced to replace the approach. As part of the system, providing a significant security increase beyond the minimum required can be rewarded with increased password lifespan before it expires. Commonly accepted wisdom is that a 3-month expiration time is acceptable. Increments can be placed at 3 month intervals for additional time, but this added time should be maxed out to around 1 year total length to avoid abuse in extreme cases. Once again, the added security should outweigh the loss of safety, which is simplified by the exponential growth of the password keyspace and linear growth of time, but it should still be monitored and implemented on a case-by-case basis.

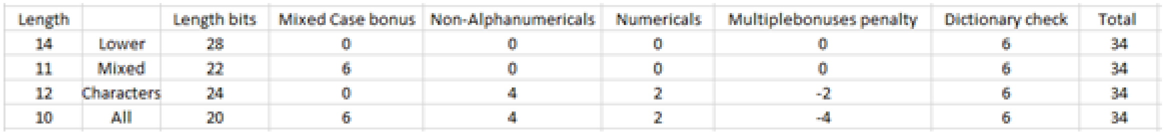
It is also important to verify that the password has not been discovered in a previous attack. Many leaked password lists have been published online, and while it may not be ethical to view them in detail, it should be embraced to check the lists for the presence of the submitted password. This knowledge could be detrimental for any attempted attacks since the leaked information is often used to execute the tactic. Another common tactic is to attempt dictionary words as passwords, along with minor variations and simple substitutions such as capitalizing the first letter or replacing the letter ‘e’ with a ‘3’. A simple dictionary check can be performed at the moment of submission and passwords should be given preference for not failing the comparison check. Those checks should also be performed with the simple substitutions in mind. Where possible, they should be reverted, such as placing all the letters in their lowercase form so that the scan is case-insensitive for easy of verification.

From these criteria, the rules for password acceptance can be built, the explicit rules for which can be stated as follows:

All passwords begin with a base value of 0 points of entropy.

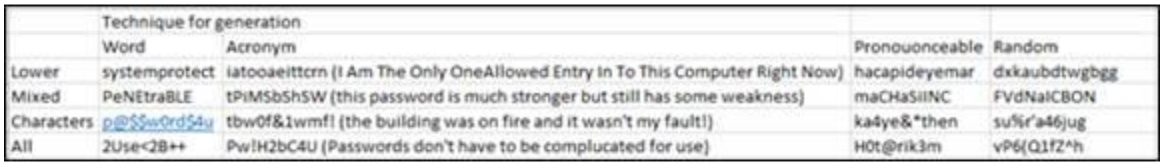
- Each Character adds 2 points of Entropy.
- Inclusion of both uppercase and lowercase characters adds 6 points of entropy.
- Inclusion of numericals in addition to alphabetic characters adds 2 points of entropy.
- Inclusion of non-alphanumeric characters adds 4 points of entropy.
    - Non-alphanumeric characters are recommended to be limited to printed keyboard characters consisting of the following set {`~!@#$%^&*()-_=+[{]}\|:;"'/?.>,<} minus any characters that could be dangerous to include in the character space.
- Inclusion of numericals in addition to alphabetic characters adds 2 points of entropy
- The bonuses for the inclusion of mixed cases, numerals and non-alphanumerics is reduced by 2 for each bonus added after the first.
    - For example: Including mixed cases and non-alphanumerics is 8 points. (6 (Cases) + 4 (Non-Alphanumerics) - 2 (multiple bonuses))
- Passing a comprehensive dictionary check adds a value of 6 points of entropy.
    - The password should be checked with a sufficiently large dictionary, at least 50,000 words. Replacing the uppercase characters with lowercase equivalents for consistency and it should not contain any of those words as a substring. Passwords found to be consisting of permutations of the username are also considered to fail this test.
- Password tests to find currently acceptable entropy thresholds should:
- Be performed on current high-performance consumer accessible hardware.
- Be performed periodically and in accordance with changes to hardware landscape.
- Be performed using all valid characters.
- A baseline minimum entropy threshold should be calculated based on a length that takes at least 6 months to complete exhaustively.
- Password lifetimes should default to three months, with an additional three months allotted per 6 points of entropy exceeded when compared to the minimum.
    - Absolute maximum allotment of a year.

Table 3
Entropy Calculation Example



Length		Length bits	Mixed Case bonus	Non-Alphanumericals	Numericals	Multiplebonuses penalty	Dictionary check	Total
14	Lower	28	0	0	0	0	6	34
11	Mixed	22	6	0	0	0	6	34
12	Characters	24	0	4	2	-2	6	34
10	All	20	6	4	2	-4	6	34

Table 4
Password Examples



	Technique for generation			
	Word	Acronym	Pronouonceable	Random
Lower	systemprotect	iatooaeittcrn (I Am The Only OneAllowed Entry In To This Computer Right Now)	hacapideyemar	dxkaubdtwgbgg
Mixed	PeNEtraBLE	tPiMSbShSW (this password is much stronger but still has some weakness)	maCHaSiINC	FVdNaICBON
Characters	p@$$w0rd$4u	tbw0f&1wmf! (the building was on fire and it wasn't my fault!)	ka4ye&*then	su%r'a46jug
All	2Use<2B++	Pw!H2bC4U (Passwords don't have to be complucated for use)	H0t@rik3m	vP6(Q1fZ^h

EXAMPLE OF USE FOR CALCULATION OF BASE VALUES

Test should be performed using whatever algorithm will be used for hashing the passwords, in this example it will be performed using SHA258 on the computer used in the previous example. For tradition and convenience, the example will begin with an initial value of 8 characters. The rules are set for the '?a' rules present in Hashcat, which includes all valid character types (uppercase, lowercase, numbers, non-alphanumeric). This causes an estimated time of approximately 35 days to calculate all possible passwords. Therefore, the password length is changed to 10 and the test is performed again which then produces an estimated result of 3 years and 91 days. This produces an acceptable threshold for a basic 3-month lifespan. The test is then repeated using entropy equivalents to confirm that the other entropy structures produced also have strong execution times, which can be seen in Table 3. This table demonstrates how a complexity requirement can be adjusted to find entropy equivalents in this system. Using a base of 10 characters with all available characters for our set, it becomes possible to calculate how many characters would be needed to reach an equivalent score using simpler character sets. For example, reducing the character set to only lowercase characters reduces the entropy by 8 points, which must then be made up using an additional 4 characters as part of the password. The table also demonstrates password groups that are entropy equivalent for our guidelines. They present equivalent levels of unpredictability, using several different methods to achieve it, such as using less characters, with a wider character set, or very small character set with a larger character total.

This allows for a wide variety of passwords to be implemented. Table 4 shows a list of example passwords generated in assorted styles and strategies. Each of the provided examples shows possible password that can be generated, all of them entropy equivalent, in several different styles. This includes traditional password/phrases, Acronym passwords (representing a phrase using specific characters to replace words), pronounceable passwords (That seek to emulate traditional words for memorability), and true randomly generated passwords. Each of these can be constructed as part of all the mentioned character sets and would all be approvable by this scheme in the case that the entropy requirements permit.

All the presented examples are performed using the assumption that all inserted passwords will pass a dictionary check. This is done so that in the case that when they do not pass a check, they will be forced to produce a stronger password, and continue to provide strong security for the password system.

PRODUCT

The product of this project was a series of algorithms that implemented the lessons learned and mitigated the weaknesses exploited in password based attacks. They follow all the outlined rules and were generated in such a way that all the observed weaknesses in password structure were at least mitigated. The primary benefit of the algorithms is versatility and scalability. The criteria for password acceptance is not tied to specific values, but to expected threats. The system can grow naturally without need to change values as hardware improves and should remain viable while attack trends hold.

The algorithms also embrace usability, users are no longer forced to adhere to the password rules, instead their passwords are evaluated on various criteria and if their combined result is satisfactory, the password is accepted. The user is given various avenues to produce additional forms of password safety and they are all beneficial. In addition to this, they are encouraged to provide passwords stronger than the average by the reward scheme of giving additional password usability lifetime.

It is important to note, however, that the algorithm presented is a template and not a hard set of rules. Implementation should be treated as an instance with changing details. Rigid adherence to those rules goes against the nature of this project and could have significant detrimental effects to its efficiency. Simple implementations can work as is, but experimentation should be encouraged for better results.

The algorithms all follow a similar structural core of calculating the strength of a given password, or a category of passwords. The calculation begins with counting the total length of the password, this number is then multiplied by 2 and becomes the base value of the tested password. From there, additional value is added based on fulfilling certain criteria. Those include:

- +6 for inclusion of both uppercase and lowercase characters.
- +2 for inclusion of numbers.
- +4 for inclusion of non-alphanumeric characters.

To prevent scaling issues, a penalty of -2 is applied to for each bonus given by this point past the first. This prevents a significant part of the value being added exclusively from bonuses.

After this point, a dictionary check is performed on the password to confirm that it is not a commonly found password or dictionary word. In the case that it does not fail the check, a final bonus of +6 is given to the password. This check is performed each time a potential password is input in to the system, as part of the approval process for the use of the password.

One of the core features of this approach is stated as scalability, and for this reason, there is also a secondary algorithm that handles calculating baseline value requirements for the passwords to be approved. This is performed by selecting an initial test value, and calculating an estimated time to exhaustively test all possible passwords in a key space of that length and a full character set of uppercase characters, lowercase characters, numbers and non-alphanumeric characters. These tests should be performed on current hardware, which should be representative of expected threats to the system. Multiple GPU clusters for large corporations and single enthusiast grade GPU systems for smaller organizations and companies for example. This produces realistic values that give users the most flexibility that can be practically offered without sacrificing safety. The goal should be to achieve at least double the desired time for password lifespan at the base value.

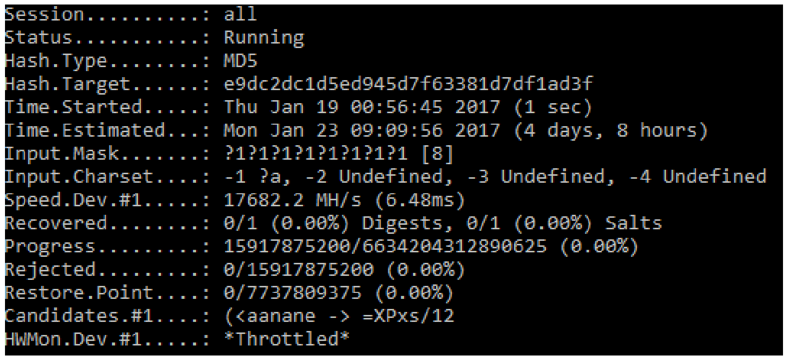
Using the test machine and settings presented in the previous sections, this proved to happen at a length of 10 characters. This should then be confirmed to be usable by performing the same test using the chosen lengths entropy equivalents in other character sets, all of which should have an execution time of a minimum satisfactory length.

Examples of these entropy equivalence calculations can seen in Table 3. These entropy equivalents shown in the table are values calculated by the author, using the rules and algorithms created for this document. The columns of which contain the possible lengths of passwords, and what requirements it must meet in order to achieve its required example value of 34. The point values of these sources can be seen in the entries of each row, with values of 0 meaning that it was not awarded a bonus for that field.

The final calculation that must be performed as part of the algorithm happens at the time of potential password submission. If the submitted password surpasses the base value required, then additional time can be given to the passwords usable lifespan as a usability tradeoff. This allows users to change their password less frequently. It also encourages stronger passwords past the minimum requirement, which provides a net gain in security, despite the increased lifespan. This is due to the exponential growth rate of password hash calculation versus the minor linear growth rate that can be given to password. As provided, the rate is that for every 2-value added to the password past the requirement, an additional 3 months be allowed for the password, to a maximum of 1 year to prevent abuse cases.

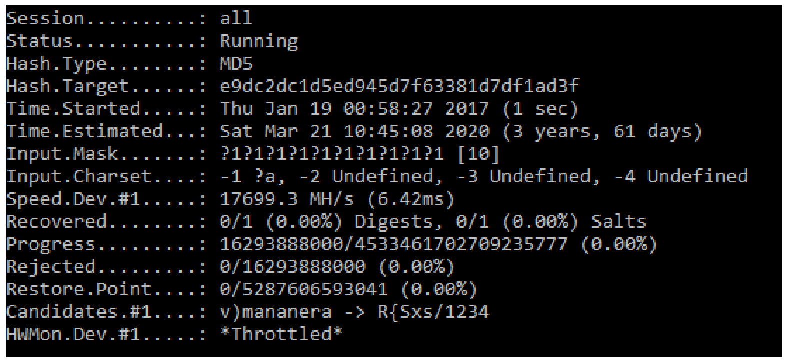
The results of implementing this system would remove many pass words that would be accepted by currently used schemes such as “password” which carries a life expectancy in the order of seconds, while also allowing new creative approaches such as “ihavemuch<3forchickentacos” which would be rejected as a weak password by some schemes for not containing a capital letter. Maintaining the length of characters to a currently relevant requirement also impacts the character space greatly, as seen in Table 1, where rapidly grows as those lengths are increased, even by small margins. Even minor increases show orders of magnitude in change for password length intervals.

This leads directly to additional burden to attackers attempting automated methods of infiltration. Conventional methods of password requirements, such as the typical minimum length of 8 with mixed cases and special character inclusion, lead to short execution times for attacks. As can be seen in Figure 2, while following the guidelines presented here result in guaranteed average execution times, such as can be seen in Figure 3. Both Figures 2 and 3 present execution times for attacks in the popular program known as Hashcat which breaks password hashes, with Figure 2 presenting the aforementioned complex 8 ruleset mentioned before and Figure 3 presenting an acceptable runtime discovered using this algorithm.



```
Session..........: all
Status...........: Running
Hash.Type........: MD5
Hash.Target......: e9dc2dc1d5ed945d7f63381d7df1ad3f
Time.Started.....: Thu Jan 19 00:56:45 2017 (1 sec)
Time.Estimated...: Mon Jan 23 09:09:56 2017 (4 days, 8 hours)
Input.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Input.Charset....: -1 ?a, -2 Undefined, -3 Undefined, -4 Undefined
Speed.Dev.#1.....: 17682.2 MH/s (6.48ms)
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 15917875200/6634204312890625 (0.00%)
Rejected.........: 0/15917875200 (0.00%)
Restore.Point....: 0/7737809375 (0.00%)
Candidates.#1....: (<aanane -> =XPxs/12
HWMon.Dev.#1.....: *Throttled*
```

Figure 2
Complex 8 Runtimes



```
Session..........: all
Status...........: Running
Hash.Type........: MD5
Hash.Target......: e9dc2dc1d5ed945d7f63381d7df1ad3f
Time.Started.....: Thu Jan 19 00:58:27 2017 (1 sec)
Time.Estimated...: Sat Mar 21 10:45:08 2020 (3 years, 61 days)
Input.Mask.......: ?1?1?1?1?1?1?1?1?1?1 [10]
Input.Charset....: -1 ?a, -2 Undefined, -3 Undefined, -4 Undefined
Speed.Dev.#1.....: 17699.3 MH/s (6.42ms)
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 16293888000/4533461702709235777 (0.00%)
Rejected.........: 0/16293888000 (0.00%)
Restore.Point....: 0/5287606593041 (0.00%)
Candidates.#1....: v)mananera -> R{Sxs/1234
HWMon.Dev.#1.....: *Throttled*
```

Figure 3
Complex 10 Runtimes

CONCLUSION

After going through the weaknesses found in current password schemes, new guidelines are recommended. The new structure is less rigid: it embraces potential sources of strength and user variances as a source of unpredictability, which allows for varied password styles and encourages users to produce stronger passwords. This can lead to a better relationship between users and their password based systems, meaning that the password validation system might not be antagonistic and complicated, but instead as a system working with the user to keep their systems secure.

Brute Force style attacks rely on the certainty that the password will eventually be discovered, but by introducing varied password length that certainty is reduced. A variety of lengths in the used passwords space, means that a wider span of possible passwords must be tested. This may bring testing of passwords outside of practical reach for password crackers.

The adoption of a variable set of password rules also embraces many styles of secure passwords that are not permissible in several current schemes. Allowing users to use measures such as sufficiently long passwords that contain only lowercase letter allows them to secure their accounts without forsaking comfort and usability for the user. The password negotiation aspect also lends itself to both security and usability. Allowing for more secure passwords to have longer usable timespans encourages the users to actively participate in the betterment of the security process instead of merely requiring them to give a minimum effort. Combined these elements lead to an improved security system that help to cover many of the current issues found in password security.

FUTURE WORK

Further study into the characteristics of human generated passwords is critical to improving security. This element has proven to be a source of significant weaknesses across a variety of password attacks. This includes further delving in to how a person's demographic information influences their password contents and structure. The previously performed study by the author [2] showed that there are unexplored elements that can influence the password creating process.

In addition, the algorithm itself can be refined on a case basis, the implementation can vary and be further developed as seen fit by the developer that plans on using it; the needs of the implementer will vary on a case-by-case basis.

REFERENCES

[1] Scarfone and S. Murugiah, "Guide to Enterprise Password Management," in *National Institute of Standards and Technology*, 2009.

[2] J. Hernandez, "A Study on the Password Habits of College Students: Length and Complexity Based on Demographics," in *Richard Tapia Celebration of Diversity in Computing*, Austin, 2016.

[3] MedicineNet. (2016, June 9). *Medical Definition of Pavlovian Conditioning* [Online]. Available: http://www.medicinenet.com/script/main/art.asp?articlekey=4801. [Accessed 2 6 2017].

[4] S. L. &. M. J. D. Brand, "Department of Defense password management guideline," in *Department of Defense*, Fort George G. Meade, 1985.