Myanmar Dictionary Based Proactive Password Checker

La Wynn Sandi, and Nyein Aye

***Abstract*—** Nowadays, passwords become extremely important and sensitive information to gain access to many online services including e-mail, networks and e-commerce applications. And the security issues relating to the password authentication become more and more important. Numerous cryptographic protocols also rely on passwords selected by users (people) for strong authentication. Passwords are an important aspect of computer security and are also the front line of protection for system and user accounts. Most of the people enjoy using easy and memorable passwords that are weak against dictionary attacks. The proposed system will analyze acceptability and strength of the selected user passwords by using proactive password checker, especially this system can resist against Myanmar language based dictionary attacks.

***Keywords*—** bloom filter, dictionary attack, proactive password checking, unicode longest matching algorithm.

I. Introduction

TODAY, there are many researches and publications in Natural Language Processing (NLP) and they are trying to develop Myanmar language based ICT applications. At present, client-server applications in Myanmar language are also still in research area (e.g. Myanmar language search engine).The leak of research and development in security approach will challenge Myanmar language research's trends and NLP research area. It will also be unavoidable to create Myanmar language based password-schemes that resist from dictionary attack. Therefore, it is necessary to develops and implement Myanmar dictionary based proactive password checker.

For different reasons, including obvious security concerns, users have to use different passwords for different systems or services, making it more difficult to remember and protect one's password. Passwords are not only critical for login identification, but also in more sophisticated service-granting systems, such as Kerberos [Neuman and Tso 1994]:, an attacker can easily mount a password-guessing assault.

Password security is an old problem. Due to the limitation of human memory, people are inclined to choose easily guessable passwords (e.g. phone numbers, birthdays, names of family or friends, or words in human languages) that might lead to severe security problems. Proactive password checking has been a common means to enforce password policies and to prevent users from choosing easily guessable passwords in the first place. When a user chooses a password, a proactive checker will determine whether his password choice is acceptable or not.

All such tests are important, but the heart of a proactive password checker has to deal with detecting membership in large dictionaries. There are two problems in this simple approach: space required to store the dictionaries, and time required to detect membership. Time is important because the user has to wait for the command prompt while the password is being checked.

Therefore, proactive password checking becomes important and it should be performed by websites before choosing a password. Nowadays, more and more Myanmar language-specific contents are being used on the Internet. This paper proposes a proactive password checking framework/design for passwords written in Myanmar language.

La Wynn Sandi is a student with University of Computer Studies, Mandalay (phone:+959254101158: email : lawynnsandi@gmail.com).

Dr. Nyein Aye is Professor with University of Computer Studies, Mandalay (email: nyeinaye@gmail.com).

II. Related Work

Bloom introduced Bloom filters in conjunction with an application to hyphenation programs [8]. Most words can be hyphenated appropriately by applying a few simple rules. Some words, said around ten percent, required a table lookup. To avoid storing all the words that could be handled via the simple rules, Bloom suggested using a Bloom filter to keep a dictionary of words that required a lookup. False positives here caused words that could be handled via the simple rules to require a lookup.

In literature there were several approaches for proactive password checking: Spafford suggested Bloom filters in the OPUS system [1]. Proactive password checking had been a common means to enforce password policies and prevented users from choossing easily guessable passwords in the first place. Proactive password checking scheme, based on second order Markov model.

Andreas Sotirakopoulos [3] showed that relating password strength to that of one's peers, while maintaining the standard visual cues, might yield certain advantages over lack of feedback or current practices. Word tokenizing [4] played a vital role in most Natural Language Processing applications. It was therefore useful to syllabify texts first. Syllabification was also a non-trivial task in Myanmar.

C.Herley, in [5], proposed that an overly restrictive password policy could be the cause for a bigger harm (particularly economic) than the harm the policy has meant to prevent. The research was depended on the system and on

user expectations, password policies couls have a severely negative impact on the security of the system instead of improving it. This led to the conclusion that usability of passwords and password creation policies might be even more important than security measured in bit strength or time needed to crack a password for an account.

M Walsvogel [7] described a class of best matching algorithms based on slicing perpendicular to the patterns and performing a modified binary search over these slices. And also analyzed their complexity and performance. Then introduce schemes that allowed the algorithm to "learn" the structure of the database and adapted itself to it. Furthermore, showed how to efficiently implement our algorithm both using general-purpose hardware and using software running on popular personal computers and workstations.

III. Password Managements

Password management is the process of defining, implementing, and maintaining password policies throughout an enterprise. Effective password management reduces the risk of compromise of password-based authentication systems to the extent possible. To protect the confidentiality, integrity, and availability of passwords so that all authorized users and no unauthorized users can use passwords successfully as needed. Integrity and availability should be ensured by typical data security controls, such as using access control lists to prevent attackers from overwriting passwords and having secured backups of password files.

Ensuring the confidentiality of passwords is considerably more challenging and involves a number of security controls along with decisions involving the characteristics of the passwords themselves. For example, requiring that passwords be long and complex makes it less likely that attackers will guess or crack them, but it also makes the passwords harder for users to remember, and thus more likely to be stored insecurely. This increases the likelihood that users will store their passwords insecurely and expose them to attackers. Password management may also be concerned about protecting the confidentiality of user identifiers, such as usernames.

Passwords are used in many ways to protect data, systems, and networks. For example, passwords are used to authenticate users of operating systems and applications such as email, labor recording, and remote access. Passwords are also used to protect files and other stored information, such as password-protecting a single compressed file, a cryptographic key, or an encrypted hard drive. In addition, passwords are often used in less visible ways; for example, a biometric device may generate a password based on a fingerprint scan, and that password is then used for authentication. This publication provides recommendations for password management, which is the process of defining, implementing, and maintaining password policies throughout an enterprise. Effective password management reduces the risk of compromise of password-based authentication systems.

IV. Password Selecting Strategy

There are many approaches to improving password security by selecting good passwords, such as user education, program-controlled password generation and reactive password checking (i.e., system administrators periodically run password cracking programs to search weak passwords), proactive password checking has been widely regarded as the best method.

A. Proactive Password Checking

User is allowed to select own password. At the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Proactive password checking it easy to block all possible weak passwords, e.g. by enforcing such a password policy, namely each password must have no less than eight characters, among which there is at least one lower case character, at least one upper-case character, at least one numerical character and at least one punctuation character, and there is no character occurring more than twice.

Nevertheless, this is impractical in real life, since passwords complying with such a policy might be too midcult to be memorized. There are always some tradeoffs between security and user convenience for password choice. As a basic assumption, proactive password checking algorithms typically do not enforce extremely strict policies but allow users to choose "good enough" passwords, though the criteria of "good enough" might vary in different circumstances.

V. Dictionary Attack

When a hacker cracks passwords, he can use the following two methods: 1) to do a dictionary attack, which tries each of a list of word and other possible weak passwords, and simple transformations such as capitalizing, prefixing, suffixing or reversing a word as a candidate until the hashed value of the candidate matches a password hash; and 2) to launch a brute force attack to search the whole key space, which is commonly huge. Hackers, however, always prefer to use dictionary attack, because it has proved to be very effective in history [6].

Current proactive password checkers are based on the dictionary attack. They check each user-chosen password candidate against a dictionary of weak passwords. If a candidate matches a dictionary item, or anyone of its variants that are generated by common transformations, then the candidate is an unacceptable password and is rejected.

In computer security, a dictionary attack is a technique for defeating a cipher or authentication mechanism to catch its decryption key or pass phrase by trying likely possibilities, such as words in dictionary(s).A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary.

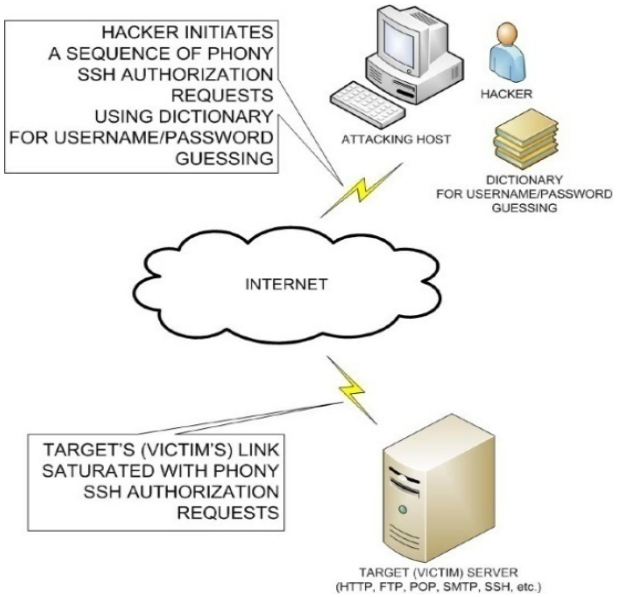

Fig. 1 Dictionary Attack

VI. Bloom Filter

A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. False positive retrieval results are possible, but false negatives are not; i.e. a query returns either "inside set (may be wrong)" or "definitely not in set". Elements can be added to the set, but not removed (though this can be addressed with a counting filter). The more elements that are added to the set, the larger the probability of false positives.

A bloom filer is based on an array of m bits($b_2,b_2,\ldots..,b_m$) that are initially set to 0.To understand how a bloom filer works, its is essential to describe how these bits are set and checked. For this purpose, k independent has functions9$h_1,h_2,\ldots,h_k$), each returning a value between 1 and m, are used. In order to "store" a given element into the bit array, each hash functions must be applied to it and, based on the return value r of each function ($r_1,r_2,\ldots.,r_k$) the bit with the offset r is set to 1. Since there are k hash function, up to k bits in there bit array set to 1. Figure is an example where m=16, k=4 and e is the element to be stored in the bit array.

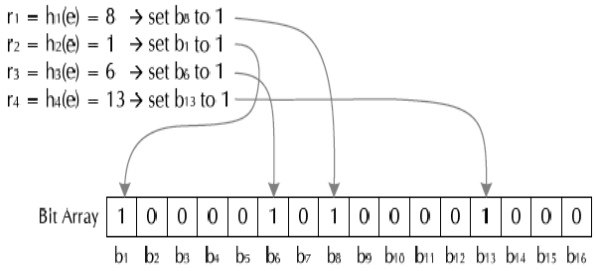

Fig. 2 Bloom Fliter

A. Applications of Bloom Filters

Bloom filters have found many applications, there are,

- Dictionary
- Databases and
- Network Application.

Bloom filters were first introduced by Bloom to keep a dictionary of words that require frequent lookup. Bloom filters were also used in early UNIX spell-checkers since a misspelled word is tolerable.

Firstly, the system accepts the input password that pass from preprocessing task. This password is marching and selecting the nearly same words in Myanmar dictionary by using Unicode longest matching .Check the sentence using grammar rules in True or False Condition. If the password is true it's hashed to use bloom filter the processes of bloom filer password checking are shown in figure.

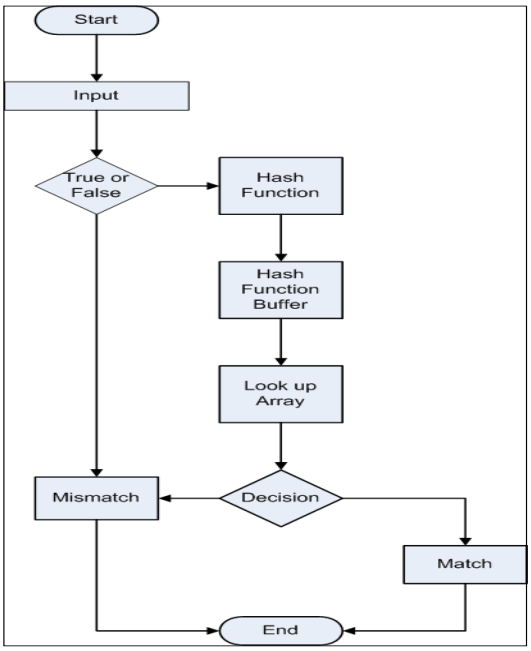

Fig. 3 Block diagram for checking a word of a string

VII. Syllable Segmentation

Syllable segmentation is the process of identifying syllable boundaries in a text.

A. Myanmar Syllables

A basic syllable consists of an initial consonant with optional medial, dependent vowels and dependent various signs.

က	= က
ကာ	= က + ာ
ကံ	= က + ံ
ကျ	= က + ျ
ကား	= က + ာ + း
ကျာ	= က + ျ + ာ
ကျား	= က + ျ + ာ + း
ကျဉ့	= က + ျ + ာ + ့
ကော်	= က + ေ + ာ + ်
ကျော်	= က + ျ + ေ + ာ + ်
ကျောင်	= က + ျ + ေ + ာ + င + ်
ကျောင်း	= က + ျ + ေ + ာ + င + ် + း

Fig. 4 Examples of Myanmar Syllables

VIII. Grammar Rules

A "grammar rule" is the right side of a production. It consists of a sequence of rule elements. The following are two of the sequence of rule elements for Myanmar Language.

A. Noun Phrase

- □□□□+□□□□ (e.g□□□□□□□□□□□ □□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□ (e.g□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□□□□)
- □□□□□+□□□□□ (e.g □□□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□□□□□)
- □□□□+□□□□□ (e.g □□□□□□□□□□ □□□□□□□□ □□□□□□□□□)
- □□□□□+□□□□ (e.g□□□□□□□□□ □□□□□□□□□ □□□□□□)
- □□□□+□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□ (e.g □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□□+□□□□□+□□□ (e.g □□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□+□□□□□ (e.g □□□□□□□□□□□)
- □□□□+□□□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□)
- □□□□+□□□□□+□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□□+□□□□□+□□□□ (e.g □□□□□□□□□□□ □□□□□□□□□□□□□□)

B. Verb Phrase

- □□□□□+□□□□□ (e.g□□□□□□□□ □□□□□□□□□□)
- □□□□□+□□□□□+□□□□□ (e.g □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□□+□□□□□+□□□□□+□□□□□ (e.g □□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□□+□□□□+□□□□□ (e.g□□□□□□□□□□□□□□□□□□□□□□□□□□□□)
- □□□□+□□□□+□□□□□+□□□□□ (e.g□□□□□□□□□□□□□□)
- □□□□+□□□□□+□□□□+□□□□□ (e.g □□□□□□□□□□□□□□□□□□□□)
- □□□□□□(□□□□□(□□□□□□□□□□□□□□□□□□□□□□□□))+□□□□ □ (e.g□□□□□□□□□□□□□□□□□)
- □□□□□+□□□□□□(□□□□(□□□□□□□□□□□□□□□□□□□□□□□□□ □□-□(□□□□□□□□□□□□□□□□□□□))

(e.g□□□)

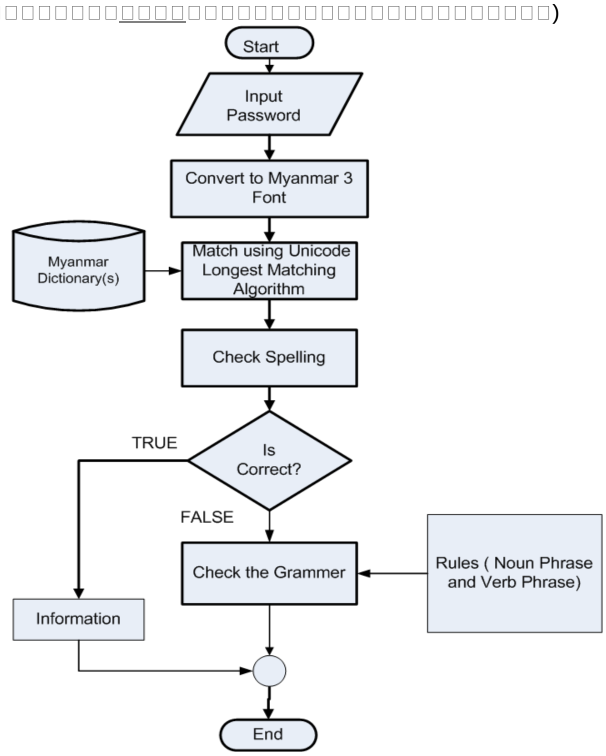

Fig. 5 Block diagram for Unicode longest Matching and grammar rules

IX. Proposed Password Checker

The proactive password checker is constructed with Bloom filter in this paper. The goal is to help the user to properly select strong and acceptable passwords, especially that can resist against Myanmar language based dictionary attacks. In the processing, the proposed system uses KANAUNG Convertor to convert any Myanmar fonts to Myanmar 3 font.

Firstly, the system accepts the input password that pass from preprocessing task. This password is marching and selecting the nearly same words in Myanmar dictionary by using Unicode longest Matching and then is hashed to use bloom filter. Check the sentence using grammar rules and is hashed to use bloom filter. Furthermore bloom filter checks the password that include in dictionary(s) or not. If the password includes in dictionary, it is rejected.

X. Conclusion

Nowadays, passwords become extremely important and sensitive information to gain access to many online services including e-mail, networks and e-commerce applications. People are inclined to use easy passwords that are weak against dictionary driven attacks. Therefore, proactive password checking becomes important and it should be performed by websites before choosing a password. All such tests are important, but the heart of a proactive password checker has to deal with detecting membership in large dictionaries. There are two problems in this simple approach: space required to store the dictionaries, and time required to

detect membership. Time is important because the user has to wait for the password is being checked.

The propose system uses longest matching to modify Bloom filter for better result and reduces the storage capacity. And the system will help the user to properly select strong and acceptable passwords, especially that can resist against Myanmar language based dictionary attacks. Acceptability and strength of the user-selected-passwords are recommended by proposed proactive password scheme.

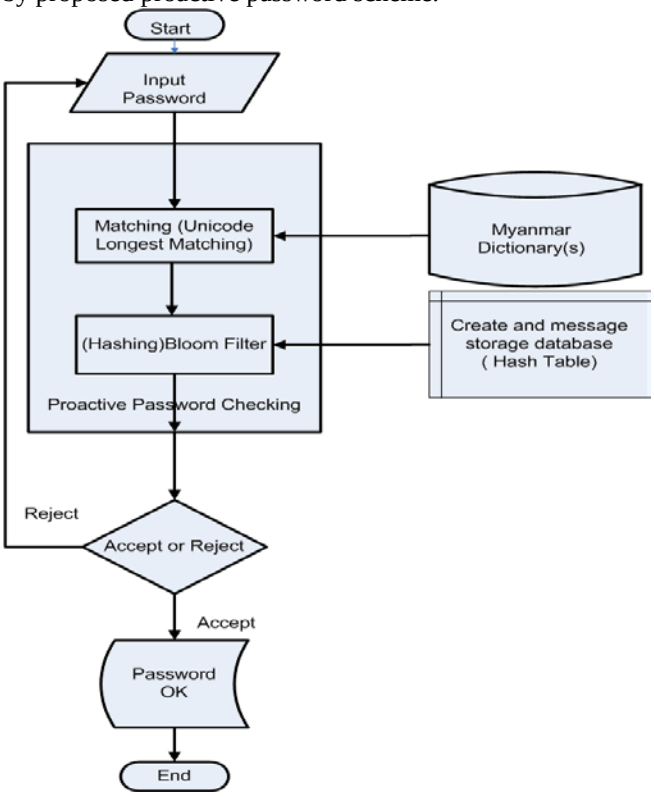

Fig. 6 Password Checking Phase of Proposed Password Checker

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] E H. Spafford, OPUS: Preventing Weak Password Choices Computers and Security,1992.

[3] H Singh Dhillon.,"Second Order Markov Model Based Proactive Password Checker", Department of Electronics and Communication Engineering, IIT Guwahati , India.

[4] H H Htay, "Myanmar Word Segmentation using Syllable level Longest Matching", Kavi Narayana Murthy,Department of Computer and Information Sciences,University of Hyderabad, India

[5] C Antognini, "Bloom Filter", Trivadis AG, Zurich, Switzerland,2008.

[6] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In NSPW '09: Proceedings of the 2009 Workshop on New Security Paradigms Workshop, pages 133–144, New York, NY, USA, 2009. ACM http://dx.doi.org/10.1145/1719030.1719050

[7] F Bergadano, B Crispo, and G Ruffo,"High Dictionary Compression for Proactive Password Checking" ACM Transactions on Information and System Security, Vol. 1, No. 1, November 1998. http://dx.doi.org/10.1145/290163.290164

[8] M Waldvogel, "Fast Longest Prefix Matching: Algorithms, Analysis, and Applications", Swiss Federal Institute Of Technology, Zurich, 2000.

[9] B. Bloom. "Space/Time Tradeoffs in Hash Coding with Allowable Errors." Communications of the ACM 13:7 (1970), 422—426. http://dx.doi.org/10.1145/362686.362692

[10] G Varghese, Network Algorithms, Lecture 4: "Longest Matching Prefix Lookups",2011

[11] Y K Thu and Yoshiyori Urano. 2006. Text entry for myanmar language sms: Proposal of 3 possible input methods, simulation and analysis. In Fourth International Conference on Computer Applications, Yangon, Myanmar, Feb.

[12] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: "A scalable wide-area web cache sharing protocol", IEEE/ACM Transactions on Networking, 8(3):281{293, June 2000.

[13] K Scarfone ,M Souppaya, "Guide to Enterprise Password Management (Draft)", Recommendations of the National Institute of Standards and Technology .

[14] R. Morris and K. Thompson, Password Security: A Case History, Communications of the ACM, Vol.22, No.11,November, 1979, pp.594-597. http://dx.doi.org/10.1145/359168.359172

[15] D.V. Klein, Foiling the Cracker: A Survey of, and Improvements to, Password Security, 2nd USENIX Unix Security Workshop, 1990, pp.5-14

[16] Hackers find new way to bilk eBay users, CNET news.com, March 25, 2002.

[17] K Ozaki and M Shimbo and M Komachi and Y Matsumoto "Using the Mutual k-Nearest Neighbor Graphs for Semi-supervised Classification of Natural Language Data" Proceedings of the Fifteenth Conference on Computational Natural Language Learning, pages 154–162,Portland, Oregon, USA, 23–24 June 2011.