Have the cake and eat it too – Infusing usability into text-password based authentication systems *

Sundararaman Jeyaraman and Umut Topkara
CERIAS and Department of Computer Sciences
Purdue University
jsr,umut@cs.purdue.edu

Abstract

Text-password based authentication schemes are a popular means of authenticating users in computer systems. Standard security practices that were intended to make passwords more difficult to crack, such as requiring users to have passwords that "look random" (high entropy), have made password systems less usable and paradoxically, less secure. In this work, we address the need for enhancing the usability of existing text-password systems without necessitating any modifications to the existing password authentication infrastructure. We propose, develop and evaluate a system that automatically generates memorable mnemonics for a given password based on a text-corpus. Initial experimental results suggest that automatic mnemonic generation is a promising technique for making text-password systems more usable. Our system was able to generate mnemonics for 80.5% of six-character passwords and 62.7% of seven-character passwords containing lower-case characters (a-z), even when the text-corpus size is extremely small (1000 sentences).

1. Introduction

Text-password based authentication schemes are a popular means of authenticating users in computer systems. The security of password based authentication systems is directly proportional to the difficulty with which an adversary can crack the passwords. A password that is difficult to crack could be intuitively thought of as a string that is not based on a dictionary word and has maximum entropy ("looks" totally random) [29, 13, 17, 38]. However, the ability to remember completely unrelated sequence of items is very limited in human beings. Hence, the more secure the password is (the greater the randomness), the more difficult it is for users to remember it. This limited ability is further taxed by the fact that a typical user has access to multiple computer systems and is advised to use a unique password for each system. The very requirements that make text-password systems secure seem to make them less user-friendly and paradoxically, less secure. Addams et al. [1] in their survey of users from various organizations, found that standard practices adopted to ensure the security of password authentication systems e.g. expiry mechanisms that force users to periodically change passwords, actually resulted in the lowering of security. For example, 50% of the interviewed users wrote down their passwords in some form – making the passwords susceptible to social engineering attacks. Also, it was found that password expiry mechanisms resulted in the users frequently making "poor" password choices.

Many alternate schemes have been proposed to achieve both security and usability in authentication systems. For example, Graphical Password schemes [10, 12, 4, 11], leverage the fact that it is easier for the users to remember pictures. However, we believe that text-password based systems will still remain prevalent in the near future for reasons such as user resistance to change and cost of modifying the existing systems. In this work, we address the need for enhancing the usability of text-password systems without necessitating any modifications. We have developed a system which could help users memorize several random passwords with ease using mnemonics. Our system is complementary to the existing text-password systems and could be used as an add-on or a helper utility. Our work is a promising first step towards reconciling, the seemingly conflicting requirements of text-password systems: security and usability.

Our system increases the memorability of random looking passwords by supplying the users with mnemonics that serve as "reminders" of the passwords. Mnemonics have been widely-used as effective memory enhancement tools

*Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

to help people remember lists of words [36]. In fact, many security-conscious users generate their passwords based on phrases that they can remember easily. For example, the sentence "The quick brown fox jumped over the lazy dog" could be used as a mnemonic to remember the password "Tqbfj01d". A recent study by Yan et al. [38] confirms the intuition that mnemonic-based passwords are memorable. But, the security of passwords generated using such an approach depends heavily on the ability of an user to come up with memorable mnemonics. It is conceivable that, as the number of passwords each user has to remember increases, the user's capability to generate memorable mnemonics becomes increasingly taxed. This could lead to insecure practices such as reusing passwords (or their minor variants) across different systems. Our system removes this dependency by relieving the user from the task of generating memorable mnemonics.

Currently, our system takes an existing text corpus and pre-computes a database of syntactic and semantic variations of its sentences, where each variation encodes a different password. Later, for any given random text-password, our system searches through the pre-computed sentence space and automatically generates a list of easier-to-remember natural language sentences that could serve as memory-aids for remembering the password.

The rest of the paper is organized as follows. In Section 2, we describe the architecture and implementation of the automatic mnemonic generation system. In Section 3, we discuss some of the issues that arose while developing our mnemonic generation system. We evaluate our system and report our results in Section 4. In Section 5, we discuss related work that address the challenge of making authentication systems more user-friendly. Finally, we discuss future work in Section 6 and conclude in Section 7.

2. Automatic mnemonic generation

The ability of human beings to remember a sequence of unrelated items is very limited [16, 2]. Hence, remembering a randomly generated password that consists of a sequence of unrelated characters is an onerous task that frequently results in users resorting to unsafe practices such as writing down their passwords [38]. However, research in cognitive psychology [7, 5] has shown that the ability to memorize and then recollect information is positively influenced by associating additional semantic content with that information. The key idea is to automatically generate and associate additional semantic content for any given password so that the additional semantic content then acts as a mnemonic device that assists the user in recalling the password. We use natural language sentences that convey some news or a story to provide the semantic content. We refer to the association between a password string and a mnemonic as encoding the password into the mnemonic. One possible way to encode a password in a mnemonic could be to represent every character in the password string by the first letter of a word in the mnemonic. For example, "The **q**uick **b**rown **f**ox **j**umped" can be used to encode the password "qbfj". We do not use stop-words to encode characters (in the previous example, the word 'The' was not used to encode any character). For expository purposes, in the rest of this section, we assume that the passwords consist entirely of lower-case Latin characters (a-z). Later in Section 3.1, we explain schemes to encode upper-case, digits and special characters.

The goal of automatic mnemonic generation is to automatically generate natural language sentences containing some news or a story to encode a given password. Instead of generating the sentences from scratch, we use a manually created corpus of natural language sentences each of which contain some news or story. For a password of a given length, if we have a large enough corpus, then it should be trivially possible to encode all possible passwords using the sentences in that corpus. However, it is difficult to obtain a large manually created corpus of sentences. Though there is a lot of text available electronically (web pages, project Gutenberg [26]), most of the sentences do not contain memorable information. For example, the sentences found in literary works obtained from project Gutenberg are mostly descriptive in nature, with very little "interesting" information to make them memorable. Moreover, even if such a corpus is available, the space requirements of such a corpus is beyond the reach of most modern day computers. For example, to support a password space of 26^8 passwords (8 character passwords with characters from a-z), we need a corpus of the size in the order of 26^8 sentences. Even if each sentence were to take only 10 bytes, such a corpus would occupy more than a *terabyte* of space.

We address these concerns by using a small core-corpus of highly memorable natural language sentences. To offset the lack of variability in a small corpus, we generate and store the semantic variants of each sentence in a compact fashion. Given a password, we check if the sentences in the core-corpus are sufficient to encode it. If not, we perform a dynamic search through the space of semantic variants to identify a variant that can encode the password. Figure 2 provides a schematic representation of the automatic mnemonic generation process. In the following sections, we describe the major components involved in the automatic generation of mnemonics.

2.1. Reuters Corpus

We use the Reuters Corpus Volume 1 (RCV1) to obtain our core-corpus. The RCV1 has over 800,000 news stories — typical of the annual English language news output of Reuters. Specifically, we use the headlines from each news

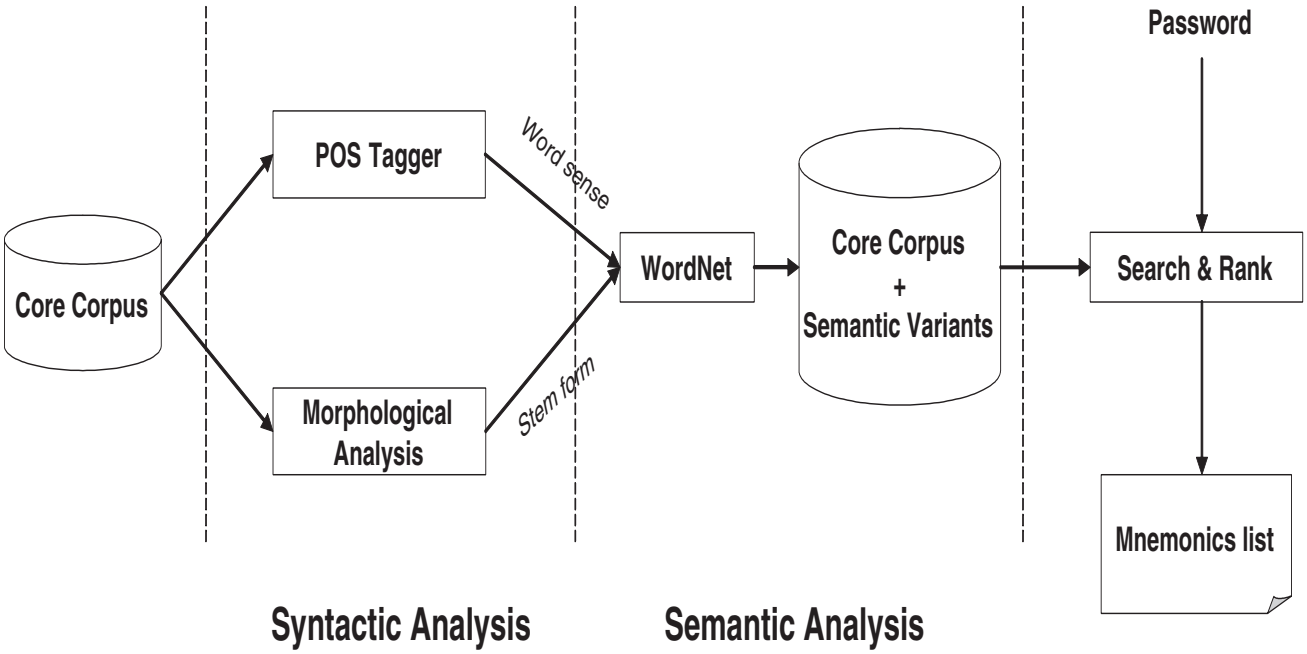

Figure 1. Schematic diagram of the automatic mnemonic generation process. The part-of-speech (POS) and stem information of each word in the core-corpus sentences is input to WordNet to produce semantic variants. For any given password, the semantic variants can then be searched for a match.

story to form our core-corpus. The headlines are particularly attractive candidates as mnemonics because:

1. They are simple in structure and hence easy to understand by even the average user.
2. They provide summaries of events and hence contain more semantic information than sentences found elsewhere.
3. They are written with the intention of catching the attention of the reader and hence are bound to be very memorable.

An example of a headline that contains memorable information is: "Egyptian plane, overshoots runway, hits Turkish taxi."

2.2. Generating semantic variants

We generate semantic variants of each sentence in the core-corpus as follows:

1. We use a sophisticated part-of-speech (POS) tagger to morphologically analyze every word in each sentence and tag it with the appropriate part-of-speech. We use the part-of-speech as a first-degree approximation of the real *sense* of the word.
2. We then use WordNet [37] and the part-of-speech information to generate semantic variants of each word.

Part of speech tagging: A vital piece of information that is necessary for generating semantic variants of a sentence is the *sense* information about each word in the sentence. Many words have different meanings or senses in different contexts. For example, the word *bank* can have the following senses: 'the river bank' or 'the Chase-Manhattan bank'. The correct sense of words is necessary to maintain semantic coherence of the generated variants. Without the knowledge of the appropriate sense, semantic variation techniques such as *synonym substitution* do not work very well. Currently, we do not distinguish between all the different senses of a word. We detect only those senses that manifest themselves as differences in parts of speech. For example, the word *butter* can be used both as a noun and a verb. We use the Stanford Log Linear Part-of-speech Tagger [30] to obtain the part-of-speech information of the words in each sentence. The part-of-speech information serves as a first-degree approximation of word sense. In addition to the part-of-speech information, we perform morphological analysis using PC-KIMMO [23] to obtain the "root" forms, tense and number information about verbs and plural nouns.

WordNet and Semantic variant space: WordNet is a lexical reference system developed by researchers at Prince-

ton University [37]. It organizes English nouns, verbs, adjectives and adverbs into synonym sets, each representing one underlying lexical concept. Different syntactic and semantic relations link the synonym sets. WordNet can be logically thought of as a labeled graph where synonym sets are the nodes and the edges represent semantic relationships. For a given word and its sense, it is possible to obtain semantically related words such as synonyms ($taxi \rightarrow cab$) , antonyms ($hit \rightarrow miss$), hyponyms ($taxi \rightarrow minicab$) and hypernyms ($taxi \rightarrow car$). It is possible to traverse further along the links to obtain a "transitive closure" of hypernyms and hyponyms (hypernyms of hypernyms, hyponyms of hyponyms, synonyms of hypernyms and so on). Let sentence S_i be a sequence of words $w_{i,1}, w_{i,2}, \cdots, w_{i,j}$. Then the semantic variant space of S_i is represented in the corpus as $W_{i,1}, W_{i,2}, \cdots, W_{i,j}$ where $W_{i,k}$ is the set of words that are semantically related to $w_{i,k}$. Figure 2 contains an example of semantically related words obtained through WordNet. By traversing through the semantic variant space, specific variants can be obtained. For example, "Zulu balloon, misses dock, kills Kirghiz ambulance" is a semantic variant of "Egyptian plane, overshoots runway, hits Turkish taxi".

2.3. Dynamic search of semantic variants

For a given password, if none of the sentences in the corpus can be used as a mnemonic, then the space of semantic variants of each sentence is searched for a possible match. Informally, a match is found if there exists at least one semantic variant that is capable of encoding the password. Formally, a match is found for a password with k characters $c_1, c_2, \cdots, c_k$, if and only if $\exists i$ such that there is a mapping $\{w_1, w_2, \cdots, w_k\} \longrightarrow \{W_{i,x} \times W_{i,x+1} \times \cdots \times W_{i,x+k-1}\}$, where $1 \leq x, x+k-1 \leq |S_i|$ and w_y encodes c_y for $1 \leq y \leq k$. For each match found, we use the tense and number information obtained using morphological analysis to regenerate the surface form of the words in the mnemonic (e.g. $\{flip, presenttense, thirdperson, singular\} \rightarrow flips$). Finally, the user chooses one mnemonic from all the listed matches.

3. Discussion

3.1. Encoding passwords in mnemonics

Apart from the first letter encoding technique used to describe the automatic mnemonic generation process, the other possible encoding schemes are:

1. nth letter encoding: This is a generalized version of the first letter technique where every character in the password string is represented by the nth letter of a word in the mnemonic. For example, the same mnemonic "T**h**e q**u**ick b**r**own f**o**x" can be used to encode the password "huro" using a *second* letter encoding. The obvious limitation for this technique is the fact that words are of limited size. For example, we cannot use a 10^{th} letter encoding in the previously mentioned mnemonic since none of the words are that long. Another limitation might be the fact that the nth letter encoding might not be as user-friendly as the first letter encoding.
2. Last letter encoding: Similar to the first letter encoding, every character in the password string is represented by the last letter of a word in the mnemonic. Using this encoding, the mnemonic "Th**e** quic**k** brow**n** fo**x**" can be used to represent the password "eknx".

Any of the aforementioned encodings can be used to encode the passwords in mnemonics and for a given password, the generated mnemonics are listed along with the encoding technique used.

3.2. Handling non lower-case characters

A challenge that arises due to the use of natural language text as mnemonics is the ability to handle characters that appear infrequently, if at all, in natural language text. e.g. Upper-case characters (A-Z), digits (0-9), special characters (\$, % etc.). In the following subsections, we propose and discuss various possible methods to encode such characters in natural language text.

3.2.1. Handling digits

Numeric digits (0-9) can be encoded in mnemonics using any combination of the following ways:

1. Digits can be treated in exactly the same fashion as the letters. For example, the password "1ioi" can be encoded using the mnemonic "In 1947 India obtained independence", where 1947 is used to encode the digit '1'. This process can be generalised to enable the same mnemonic to be able to encode multiple passwords. If we replace the example sentence with "In NUMBER India obtained independence", then the same sentence can be used to encode any password conforming to the regular expression $[0-9]ioi$. If the password is "2ioi", the mnemonic could be "In 2000 India obtained independence". This is an example of a semantic variation. However, WordNet does not handle numbers. Hence, we perform the tagging and variant generation for numbers on our own.
2. Lower-case characters can be overloaded to represent digits. A scheme similar to what is popularly referred to as "leet speak" can be used. For example, '0' is 'o', '1' is 'l', '3' is 'e' and so on. We discuss the limitations of overloading in Section 3.2.4.

```
Egyptian    {Algerian, Angolan, Basotho, Bantu, Zairese, Zimbabwean, Zulu}
plane       {airplane, autogiro, drone, glider, helicopter, orthopter, warplane}
overshoots  {miss, shoot, overshoot, undershoot}
runway      {platform, auction_block, bandstand, catwalk, dais, dock}
hit         {play, foul, ground_out, toe, snap, kill, drive, hit, launch, loft}
Turkish     {Azerbaijani, Kazak, Tatar, Uzbek, Uighur, Yakut, Kirghiz}
taxi        {cab, hack, taxi, taxicab,  minicab, car, automobile, machine}
```

Figure 2. Examples of semantic relatives of the words in the sentence "Egyptian plane, overshoots runway, hits Turkish taxi" obtained from WordNet.

3. Instead of using a single letter to represent a digit, a whole word can be used instead. This scheme is similar to the classical "pegword" mnemonic device which uses words that rhyme with digits. For example, '1' is 'bun', '2' is 'shoe', '3' is 'tree' and so on.

3.2.2. Handling Upper-case letters

We use the first letters of proper nouns found in the natural language text to encode upper-case letters. For example, "Paul chased the dog" is a mnemonic for the password "Pctd". To be able to accomodate occurences of upper-case letters in different positions, we use semantic tagging and variant generation similar to the tagging used in handling digits. For example, the previous mnemonic is replaced in the corpus with "NAME chased the dog" and a list of all names is maintained separately. So now, the mnemonic could be used to encode any password satisfying the regular expression $[A-Z]ctd$.

3.2.3. Handling special characters

There are 32 special characters that are printable in the ASCII characters set, almost none of which appear in natural language text (except those used for punctuation). So, we handle the special characters by overloading the lower-case characters. If we restrict the number of special characters that appear in the password to less than 26, then for each special character, a unique mapping to a lower-case character can be found. On the other hand, if we want to allow all 32 special characters, then we need to use a sequence of two lower-case characters to represent a single special character e.g., '%' is 'aa', '#' is 'bb'.

3.2.4. Overloading lower-case characters

A consistent theme that arises during attempts to encode characters that do not frequently appear in natural language text such as digits and special characters, is that of overloading. Overloading a lower-case character to represent digits and special characters introduces an additional level of encoding that destroys the one-to-one mapping between a mnemonic and the actual password. For example, the mnemonic "The asinine brown fox" could encode both "abf" or "4bf" (if '4' is encoded as 'a' as discussed in section 3.2.1) or "]f" (if ']' is encoded as 'ab'). In the worst case, a user who just remembers the mnemonic and the encoding scheme might have to try all the different possible password candidates. We are of the opinion that after a few initial trials during which the user might have to try all possible candidates, the user will be able to recollect the overloaded information precisely. We also believe that the cost of having to remember which letter has been overloaded is very minimal when compared to actually remembering the password itself without the aid of the mnemonics.

3.3. Personalisation of the corpus

Though the core-corpus contains highly memorable sentences, it could be made more memorable by personalising the mnemonics generated according to the user. For example, an user who is a soccer fan, might find a soccer related headline to be more memorable than a headline about the latest stock exchange news. The RCV1 corpus was designed with text-classification applications in mind. Every headline and newstory comes tagged with information about one or more of the categories that it belongs to. This makes personalisaton extremely simple.

3.4. Ranking the mnemonics

For many passwords, there are multiple candidates as mnemonics. There is no general way to rank them. One possible way to rank candidates that are among the semantic variants, is to use a *semantic distance metric* that measures how close they are semantically related to the original sentences they are derived from. The semantically closer relatives are ranked higher. The intuition behind using the se-

mantic distance metric is as follows: The original sentences are the most desirable due to their memorability. The further removed the relation between a semantic variant and the original sentence is, the greater the chance that the variant does not retain the memorability of the original sentence.

4. Evaluation

The two properties that we desire in our automatic mnemonic generator(AMG) are *memorability* and *coverage*. Memorability refers to the ease with which the generated mnemonics can be remembered and recollected. We depend on the appealing nature of the sentences in our core-corpus for the memorability of the generated mnemonics. Coverage refers to the number of passwords for which our system can generate mnemonics. The coverage of AMG depends on a variety of factors such as the length of the password, the size of the corpus, and the encoding technique. To measure coverage, we define a ratio called Coverage Ratio (CR). Let n be the maximum number of characters in the password and let S be the alphabet from which the characters of the password are obtained. For a given n, the maximum number of passwords that can be generated using S, $N = \sum_{i=1}^{n} |S|^i$. For a given n and S, the ratio CR is defined as m/N where m is the maximum number of passwords for which AMG can generate mnemonics.

To measure CR, for a given n and S, we need to be able to measure m. m is dependent on factors such as the size of the corpus, the nature of the corpus, the size of the WordNet. This makes it difficult to measure m accurately without having to exhaustively enumerate all the passwords that our system can support. Exhaustive enumeration is computationally expensive for large values of m and n (it is equivalent to a cracker using brute-force to crack a password). Hence, we resort to *sampling* to obtain an estimate of CR. We randomly generate k passwords and test whether AMG can generate mnemonics for the generated passwords. The ratio k'/k is then a probabilistic estimate of CR, where k' is the number of passwords for which AMG is successful in generating mnemonics.

We perform two sets of experiments, one for $n = 6$ and another for $n = 7$. For both experiments, we generate passwords that contain only lower-case characters. In each experiment, we randomly generate a sample of $k = 1000$ passwords and obtain k' using three different encodings (1st letter, 2nd letter and 3rd letter) to obtain mnemonics. The Best-coverage encoding is the union of the results of the three encoding schemes. The size of the core-corpus used was extremely small – 1000 sentences. Figure 3 plots the measure of success of AMG in generating mnemonics for both the experiments. Even with such an extremely small core-corpus, our system is able to achieve a coverage ratio of 80.5% for six-character passwords and 62.7% for seven-character passwords.

5. Related Work

In this is section, we discuss related work that either share or could be potentially used towards realizing the same broad goal as ours i.e. making password system more usable.

5.1. Graphical Passwords

The lack of usability in text password systems and the resultant reduction in security has led researchers to propose alternate authentication schemes such as graphical passwords. Graphical passwords leverage the fact that pictures are easier to remember than text [18, 6, 20]. In general, graphical password schemes can be divided into two categories: recognition-based and recall-based schemes.

Recognition-based schemes such as $Passfaces^{TM}$ [9, 32] and Deja-Vu [11, 24] leverage the proficiency of human beings in recognizing previously seen images [19, 11]. However, as Davis et. al. [10] found out, user choice in such systems could lead to passwords that have much lower entropy than the theoretical optimum, and in some cases, are highly correlated with the race and gender of the users.

Recall-based schemes require users to precisely recall a sequence of strokes or points on pictures. For example, Jermyn et al. [12] describe a scheme where a password consists of a series of lines drawn by the user using an input device (for example, stylus). By decoupling the temporal order in which the lines are drawn from their positions, they observe that they can produce interesting password schemes that are at least as secure as the text passwords while being easier to remember.

Graphical passwords are a promising alternative for text-based password systems in the future. However, as studies by Davis et al. and Thorpe et al. [33, 34] suggest, graphical password technologies are not mature enough to be deployed immediately. Text-password systems with enhanced usability (using systems such as ours), could serve as a viable alternative until such technologies become ready for widespread deployment.

5.2. Password Generators

Commercially available tools such as $PasswordGenerator^{TM}$ [22], [21] use a series of ad-hoc techniques to help users generate secure and memorable passwords. As far as we know, there has been no study done on the security and the memorability of the passwords generated by these tools. Also, there has been no study to test the ability of such tools to generate a large number of passwords. The ability to cover a large password

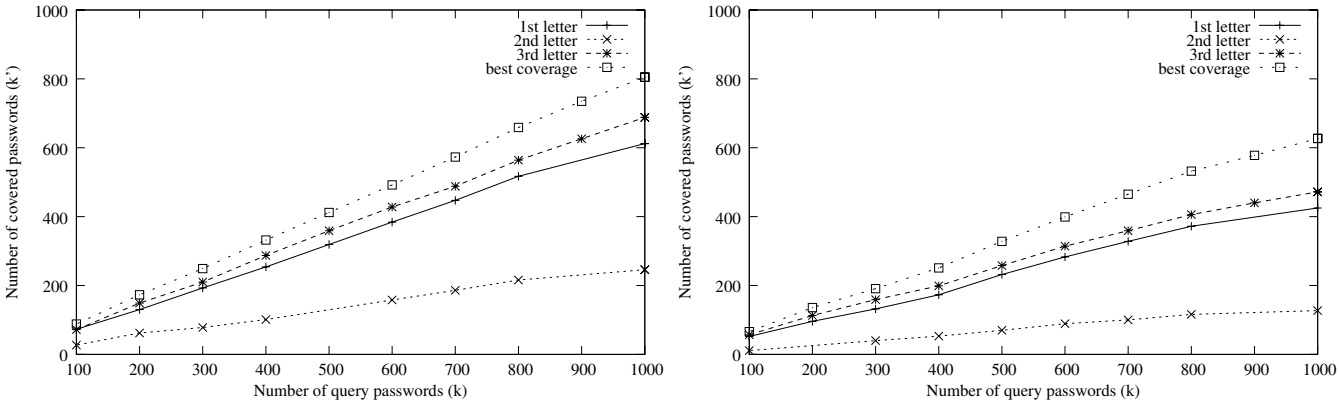

Figure 3. Coverage plots for six and seven character passwords.

space is important because otherwise a cracker could brute-force through all the possible passwords generated by such a tool.

5.2.1. Mnemonic Generators

Research by Raskin et. al. [3] and the mnemonic-password generator from [31] are the closest to our work. As far as we are aware, those are the only two attempts, other than our work, at generating mnemonics by leveraging Natural Language Processing algorithms. Raskin et. al. [3] try to automatically generate "humorous" verse to help remember passwords. While humorous verse is a potentially effective way to remember passwords, automatic generation of humorous verse is a daunting challenge. Existing tools for automatic humor generation do not scale well enough to encode a large password space. For example, Raskin's system can generate mnemonics for passwords that consist of eight characters chosen from a Latin character set (a-z). Our system can be easily extended to handle arbitrary password length and is capable of handling passwords with characters from an arbitrarily large character-set.

Very little public documentation is available for [31]. Their system automatically generates a series of <password, mnemonic> tuples some of which could be appealing to an user. Similar to our system, they maintain a set of sentence templates internally. e.g., "<noun>'s <verb> <adv>, <conj> <noun>'s <verb> <adv>". For each token in the set of templates, a set of words are maintained internally as possible mnemonics. For example, <noun> could be associated with Mark, Dan, Gene, Snake etc., Every time a user requests a password to be generated, the system randomly picks a sentence template and for each token in the template, randomly chooses a word associated with the token. Each word acts as a mnemonic for a character in the generated password. For example, the template "<person>'s <noun> <verb>s <person>'s <adj> <noun>" could be used to generate the sentence "Mark's canary illustrates Dylan's ninth haddock" which encodes a password "MciD9h" [25]. The following differences highlight the advantages of our system over [31]:

- **Sentence templates:** Our system automatically generates the sentence templates from an existing text corpus whereas [31] depends on a manually created template corpus. Using an existing corpus has the following advantage: The generated mnemonics look more "real" and have a greater variety as opposed to the "manufactured" feel of the sentences generated by [31].
- **Semantics of the generated mnemonics:** By virtue of using a text-corpus, the mnemonics generated by our system have richer semantic content than those generated by [31].
- **Extensibility:** Because of the ability to generate syntactic and semantic variants of the sentences from the text-corpus, our system can be easily extended to generate memorable mnemonics for a larger set of passwords and for passwords of arbitrary length.

5.3. Automatic Story Generation Systems

Automatic story and prose generation are classical problems studied in the fields of Artificial Intelligence and natural language generation respectively. Stories generated by automatic story telling machines could be potentially used as mnemonics. However, existing automatic story generation tools [15, 14, 8], were not designed with our needs in mind and hence are not directly applicable in our context because of the following reasons:

- **Interactivity:** Most of the story generation systems are interactive and are not completely automatic. We need to be able to automatically generate stories.

- **Lack of semantic variability:** The story generation systems do not generate plot-lines for the stories by themselves from scratch. The plot-lines and most of the information needed for narrating a story is available to the generation systems in the form of a knowledge base. Unfortunately, all the systems we know of are research prototypes that were built to demonstrate the effectiveness of the story generation procedure itself. Hence their knowledge bases are often extremely limited. Because of this limitation, the stories generated are often repetitive and "mechanical" in nature. This not only severely affects the memorability of the generated stories, but also limits the number of unique stories generated and hence the number of passwords covered.

6. Future Work

The most appealing feature of our core-corpus is that the sentences contain appealing semantic information that make them highly memorable. Semantic variants of a sentence might be less memorable than the original sentence. However, syntactic variants are very attractive since they retain the semantic content and hence the same level of memorability. Similar to the generation of semantic variants, it is possible to generate syntactic variants of every sentence using Natural Language Generation (NLG) techniques. Once a sentence is parsed and is represented using an abstract text representation, the flexibility of the abstract representation allows us to generate many syntactic variants of the same semantic content using natural language generation tools such as RealPro [27].

While generating semantic variants of sentences in our core-corpus, we rely on part of speech information as indicators of senses of the words. We would like to incorporate the ability to perform word sense disambiguation at a finer granularity than part of speech. We plan to leverage existing tools such as [28].

The headlines used in the core-corpus are short in nature. Hence, the length of the passwords that can be encoded using the headlines is limited. In the future, we would like to explore ways to generate memorable natural language text that can encode longer passwords.

The most critical issue we would like to investigate is the memorability of the mnemonics generated by our system. Unlike coverage, there are no objective measures for memorability. Hence, we plan to conduct an elaborate usability study to answer the following questions:

- How much longer does a user remember a random password with the aid of the mnemonics generated by our system?
- How many such passwords can a user remember simultaneously using our mnemonics?
- What is the impact of encoding techniques (the nth letter encoding, overloading special characters) on the effectiveness of the mnemonics?

7. Conclusion

In this paper, we propose an automatic mnemonic generating system to make text-password systems more usable. Our system helps users remember crack-resistant passwords by automatically generating mnemonics. We discuss the issues that arise while trying to encode passwords using mnemonics. We evaluate our system and discuss potential future improvements. Our initial results indicate that automatic mnemonic generation is a very promising approach for infusing usability into text-password systems.

Acknowledgments

We thank Prof. Mike Atallah, Mercan Topkara and all the anonymous reviewers. Their feedback helped us improve the quality of this paper.

References

[1] A. Addams and M. A. Sasse. "Users are not the enemy," *In Communications of the ACM,* Volume 42 , Issue 12, 1999, pages 40-46.

[2] John R. Anderson and Christian Lebiere. *The Atomic Components of Thought*

[3] Mikhail J. Atallah, Craig J. McDonough, Victor Raskin, Sergei Nirenburg. "Natural language processing for information assurance and security: an overview and implementations," *Proceedings of the 2000 workshop on New security paradigms,* 2000.

[4] G. E. Blonder. "Graphical password." US Patent 5559961, Lucent Technologies, Inc., Murray Hill, NJ, August 30, 1995.

[5] G. H. Bower, "Analysis of a mnemonic device," In *American Scientist,* 58, 496-510.

[6] G. H. Bower, M. B. Karlin, and A. Dueck. "Comprehension and memory for pictures." *Memory and Cognition,* 2:216-220, 1975.

[7] G. H. Bower and J. S. Reitman, "Mnemonic elaboration in multilist learning," In *Journal of Verbal Learning and Verbal Behavior,* 11, 478-485.

[8] S. Bringsjord D. Ferrucci, "Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine." Lawrence Erlbaum Associates, 1999.

[9] H. Davies. "Physiognomic Access Control," *Information Security Monitor,* vol. 10m no. 3, 1995m pp 5-8.

[10] D. Davis, F. Monrose and M. K. Reiter, "On User Choice in Graphical Password Schemes," *In Proceedings of the 13th UNIX Security Symposium,* August 2004.

[11] R. Dhamija and A. Perrig. "Deja Vu – A User Study: Using Images for Authentication," *In Proceedings of the 9th UNIX Security Symposium,* August 2000.

[12] I. Jermyn, A. Mayer, F. Monrose, M. Reiter and A. Rubin. "The Design and Analysis of Graphical Passwords," *In Proceedings of the 8th UNIX Security Symposium,* August 1999.

[13] D. Klein. "Foiling the cracker: A survey of, and improvements to, password security," *In Proceedings of the 2nd USENIX Security Workshop,* August 1990.

[14] H. Liu and P. Singh, "MAKEBELIEVE: Using commonsense knowledge to generate stories." In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, (pp. 957-958).

[15] J. Meehan. "TALE-SPIN and Micro TALE-SPIN." In Roger C. Schank and Christopher K. Riesbeck (Eds.), *Inside computer understanding.* Hillsdale, NJ: Erlbaum, 1981.

[16] G.A. Miller. "The Magical Number Seven, Plus or Minus Two." *Psychological Review,* vol. 63, 1956, pages 81-87.

[17] R. Morris and K. Thompson. "Password security: A case history," *Communications of the ACM,* 22(11):594 597, November 1979.

[18] D. L. Nelson, U. S. Reed, and J. R. Walling. "Picture superiority effect," *Journal of Experimental Psychology: Human Learning and Memory,* 3:485 497, 1977.

[19] J. Nielsen. *Usability Engineering.* Academic Press. 1993.

[20] A. Paivio, T. B. Rogers, and P. C. Smythe. "Why are pictures easier to recall than words?" Psychonomic Science, 11:137-138, 1968.

[21] Password Generator. Available at: http://www.alphalink.com.au/šergeb/easy-to-remember-passwords.html

[22] Password Generator 2005. Available at: http://www.diplodock.com/Products/PasswordGenerator/default.aspx

[23] PC KIMMO - A morphological parser. Available at: http://www.sil.org/pckimmo/

[24] A. Perrig and D. Song. "Hash visualization: A new technique to improve real-world security." *In Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce (CryTEC '99),* 1999.

[25] Personal Communication.

[26] Project Gutenberg. Available at: http://www.gutenberg.org/

[27] RealPro. Available at: http://www.cogentex.com/technology/realpro/index.shtml

[28] Senseclusters. Available at: http://www.d.umn.edu/ tpederse/code.html

[29] E. Spafford. "Observing Reusable Password Choices," *In Proceedings of the 3rd UNIX Security Symposium,* pages 299–312, Berkeley, CA, September 1992. Usenix Association.

[30] Stanford Log-linear Tagger. Available at: http://www-nlp.stanford.edu/software/tagger.shtml

[31] The password mnemonic generator. Available at: http://www.aber.ac.uk/cgi-bin/user/syswww/gw/mnemonic

[32] *The Science Behind Passfaces.* Revision 2, Real User Corporation, September 2001. http://www.realuser.com/published/ScienceBehindPassfaces.pdf.

[33] J. Thorpe and P. C. van Oorschoti. "Graphical Dictionaries and the Memorable Space of Graphical Passwords," *In Proceedings of the 13th USENIX Security Symposium,* 2004.

[34] J. Thorpe and P. C. van Oorschoti. "Towards Secure Design Choices for Implementing Graphical Passwords," In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC),* 2004.

[35] US Dept. of Defense, "Password Management Guideline," CSC-STD-002-85, 1985.

[36] *Wikipedia entry for 'mnemonic',* http://en.wikipedia.org/wiki/Mnemonic

[37] Wordnet. Available at: http://wordnet.princeton.edu/

[38] Jeff Yan, Alan Blackwell, Ross Anderson and Alasdair Grant. "Password memorability and security: empirical results," *Security & Privacy Magazine, IEEE,* Volume 2, Issue 5, 2004, pages 25-31.