Mnemonic Password Formulas

Remembering Secure Passwords

May, 2007

I)ruid, C^{2}ISSP
<druid@caughq.org>
http://druid.caughq.org

Abstract

The current information technology landscape is cluttered with a large number of information systems that each have their own individual authentication schemes. Even with single sign-on and multi-system authentication methods, systems within disparate management domains are likely to be utilized by users of various levels of involvement within the landscape as a whole. Due to this complexity and the abundance of authentication requirements, many users are required to manage numerous credentials across various systems. This has given rise to many different insecurities relating to the selection and management of passwords. This paper details a subset of issues facing users and managers of authentication systems involving passwords, discusses current approaches to mitigating those issues, and finally introduces a new method for password management and recalls termed Mnemonic Password Formulas.

Contents

Chapter 1

The Problem

1.1 Many Authentication Systems

The current information systems landscape is cluttered with individual authentication systems. Even though many systems existing in a distinct management domain utilize single sign-on as well as multi-system authentication mechanisms, multiple systems within disparate management domains are likely to be utilized regularly by users. Even users at the most casual level of involvement in information systems can be expected to interface with a half a dozen or more individual authentication systems within a single day. On-line banking systems, corporate intranet web and database systems, e-mail systems, and social networking web sites are a few of the many systems that may require their own method of user authentication.

Due to the abundance of authentication systems, many end users are required to manage the large numbers of passwords needed to authenticate with these various systems. This issue has given rise to many common insecurities related to selection and management of passwords.

In addition to the prevalence of insecurities in password selection and management, advances in authentication and cryptography assemblages have instigated a shift in attack methodologies against authentication systems. While recent headway in computing power have made shorter passwords such as six characters or less (regardless of the complexity of their content) vulnerable to cracking by brute force[5], common attack methodologies are moving away from cryptanalytic and brute force methods against the password storage or authentication system in favor of intelligent guessing of passwords such as. This intelligent guessing might involved optimized dictionary attacks and user context guesses, attacks against other credentials required by the authentication system such as key-cards and password token devices, and attacks against the interaction between the user and the systems themselves.

Due to all of the aforementioned factors, the user's password is commonly the weakest link in any given authentication system.

1.2 Managing Multiple Passwords

Two of the largest problems with password authentication relate directly to the user and how the user manages passwords. First, when users *are not* allowed to write down their passwords, they generally will choose easy to remember passwords which are usually much easier to crack than complex passwords. In addition to choosing weaker passwords, users are more likely to re-use passwords across multiple authentication systems.

Users have an inevitably difficult time memorizing assigned random passwords[5] and passwords of a mandated higher level of complexity chosen themselves. When allowed, they may write down their passwords in an insecure location such as a post-it note stuck to their computer monitor or on a note pad in their desk. Alternatively, they may store passwords securely, such as a password encrypted file within a PDA. However, a user could just as easily lose access to the password store. The user may forget the password to the encrypted file, or the PDA could be lost or stolen. In this situation, the end result would require some administrative interaction in the form of issuing a password reset.

1.3 Poor Password Selection

When left to their own devices, users generally do not choose complex passwords[5] and tend to choose easy to crack dictionary words because they are easy to remember. Occasionally an attempt will be made at complexity by concatenating two words together or adding a number. In many cases, the word or words chosen will also be related to, or within the context of, the user themselves. This context might include things like a pet's name, phone number, or a birth date.

These types of passwords require much less effort to crack than a brute-force trial of the entire range of potential passwords. By using an optimized dictionary attack method, common words and phrases are tried first which usually leads to success. Due to the high success rate of this method, most modern attacks on authentication systems target guessing the password first before attempting to brute-force the password or launch an in-depth attack on the authentication system itself.

1.4 Failing Stupid

When a user cannot remember their password, likely because they have too many passwords to remember or the password was forced to be too complex for them to remember, many authentication systems provide a mechanism that the author has termed *"failing stupid."*

When the user "fails stupid," they are asked a reminder question which is usually extremely easy for them to answer. If answered correctly, users are presented with an option to either reset their password, have it e-mailed to them, or perform some other password recovery method. When this type of recovery method is available, it effectively reduces the security of the authentication system from the strength of the password to the strength of a simple question. The answer to this question might even be obtainable through public information.

1.4.1 Case Study: Paris Hilton Screwed by Dog

A well publicized user context attack[3] was recently executed against the Hollywood celebrity Paris Hilton in which her cellular phone was compromised. The account password recovery question that she selected for use with her cellular provider's web site was "What is your favorite pet's name?" Many fans can most likely recollect from memory the answer to this question, not to mention fan web sites, message boards, and tabloids that likely have this information available to anyone that wishes to gather it. The attacker simply "failed stupid" and reset Hilton's online account password which then allowed access to her cellular device and its data.

Chapter 2

Existing Approaches

2.1 Write Down Passwords

During the AusCERT 2005 information security conference, Jesper Johansson, Senior Program Manager for Security Policy at Microsoft, suggested[2] reversing decades of information security best practice of not writing down passwords. He claimed that the method of password security wherein users are prohibited from writing down passwords is absolutely wrong. Instead, he advocated allowing users to write down their passwords. The reasoning behind his claim is an attempt at solving one of the problems mentioned previously: when users are not allowed to write down their passwords they tend to choose easy to remember (and therefore easy to crack) passwords. Johansson believes that allowing users to write down their passwords will result in more complex passwords being used.

While Mr. Johansson correctly identifies some of the problems of password security, his approach to solving these conundrums is not only short-sighted, but also noncomprehensive. His solution solves users having to remember multiple complex passwords, but lso creates the aforementioned insecure scenarios regarding written passwords which are inherently physically less secure and prone to require administrative reset due to loss.

2.2 Mnemonic Passwords

A mnemonic password is a password that is easily recalled by utilizing a memory trick such as constructing passwords from the first letters of easily remembered phrases, poems, or song lyrics. An example includes using the first letters of each word in a phrase, such as: "Jack and Jill went up the hill," which results in the

password "JaJwuth". For mnemonic passwords to be useful, the phrase must be easy for the user to remember.

Previous research has shown[5] that passwords built from phrase recollection like the example above yield passwords with complexity akin to true random character distribution. Mnemonic passwords share a weakness with regular passwords in that users may reuse them across multiple authentication systems. Such passwords are also commonly created using well known selections of text from famous literature or music lyrics. Password cracking dictionaries have been developed that contain many of these common mnemonics.

2.3 More Secure Mnemonic Passwords

More Secure Mnemonic Passwords[1] (MSMPs), are passwords that are derived from simple passwords which the user will remember with ease, however, they use mnemonic substitutions to give the password a more complex quality. "Leet-speaking" a password is a simple example of this technique. For example, converting the passwords "beerbash" and "catwoman" into leet-speak would result in the passwords "b33rb4sh" and "c@w0m4n", respectively.

A unique problem of MSMPs is that not all passwords can be easily transformed which limits either the choice of available passwords or the password's seemingly complex quality. MSMPs also rely on permutations of an underlying dictionary words or sets of words which *are* easy to remember. Various cracking dictionaries have been developed to attack specific methods of permutations such as the "leet-speak" method mentioned above. As with mnemonic passwords, these passwords might be reused across multiple authentication systems.

2.4 Pass Phrases

Pass phrases[4] are essentially what is used as the root of a mnemonic password. They are easier to remember and much longer which results in a password being much more resilient to attack by brute force. Pass phrases tend to be much more complex due to the use of upper and lower case characters, white-space characters, as well as special characters like punctuation and numbers.

However, pass phrases have their own sets of problems. Many authentication systems do not support lengthy authentication tokens, thus resulting in pass phrases that are not consistently usable. Like the aforementioned methods, the same pass phrase may be reused across multiple authentication systems.

Chapter 3

Mnemonic Password Formulas

3.1 Definition

A *Mnemonic Password Formula*, or MPF, is a memory technique utilizing a predefined, memorized formula to construct a password on the fly from various context information that the user has available.

3.2 Properties

Given a well designed MPF, the resultant password should have the following properties:

1. A seemingly random string of characters
2. Long and very complex, therefore difficult to crack via brute force
3. Easy to reconstruct by a user with knowledge of only the formula, themselves, and the target authentication system
4. Unique for each user, class of access, and authenticating system

3.3 Formula Design

3.3.1 Syntax

For the purposes of this paper, the following formula syntax will be used:

- $< X >$: An element, where $< X >$ is meant to be entirely replaced by something known as described by X.
- | : When used within an element's angle brackets ($<$ and $>$), represents an OR value choice.
- All other characters are literal.

3.3.2 A Simple MPF

The following simple formula should be sufficient to demonstrate the MPF concept. Given the authenticating user and the corresponding authenticating system, a formula like that shown in the following example could be constructed. This example formula contains two elements: the user and the target system identified either by hostname *or* the most significant octet of the IP address.

$$< user >! < hostname|lastoctet >$$

The above MPF would yield such passwords as:

- "druid!neo" for user druid at system neo.jpl.nasa.gov
- "intropy!intropy" for user intropy at system intropy.net
- "thegnome!nmrc" for user thegnome at system nmrc.org
- "druid!33" for user druid at system 10.0.0.33

This simple MPF schema creates fairly long, easy to remember, passwords that contain a special character. However, it does not yield very complex passwords. A diligent attacker may include the target user and hostname as some of the first combinations of dictionary words used in a brute force attack against the password. Due to the fact that only the hostname or last octet of the IP address is used as a component of the schema, passwords may not be unique per system. If the same user has an account on two different web servers, both with hostname "www", or two different servers with the same last address octet value within two different sub-nets, the resultant passwords will be identical. Finally, the passwords yielded are variable in length and may not comply with a given systems password length policies.

3.3.3 A More Complex MPF

By modifying the simple MPF above, complexity can be improved. Given the authenticating user and the authenticating system, an MPF with the following components can be constructed:

$$< u >! < h|n > . < d, d, ...|n, n, ... >$$

The more complex MPF contains three elements: $< u >$ represents the first letter of the username, $< h|n >$ represents the first letter of the hostname *or* first number of the first address octet, and $< d, d, ...|n, n, ... >$ represents the first letters of the remaining domain name parts *or* first numbers of the remaining address octets, concatenated together. This MPF also contains another special character in addition to the exclamation mark, the period between the second and third element.

The above MPF would yield such passwords as:

- "d!n.jng" for user druid at system neo.jpl.nasa.gov
- "i!i.n" for user intropy at system intropy.net
- "t!n.o" for user thegnome at system nmrc.org
- "d!1.003" for user druid at system 10.0.0.33

The modified MPF contains two special characters which yields more complex passwords, however, the passwords are still variable length and may not comply with the authenticating system's password length policies. The example MPF is also increasing in complexity and may not be easily remembered.

3.3.4 Design Goals

The ideal MPF should meet as many of the following design goals as possible:

1. Contain enough elements and literals to always yield a minimum password length
2. Contain enough complex elements and literals such as capital letters and special characters to yield a complex password
3. Elements must be unique enough to yield a unique password per authenticating system
4. Must be easily remembered by the user

3.3.5 Layered Mnemonics

Due to the fact that MPFs can become fairly complex while attempting to meet the first three design goals listed above, a second layer of mnemonic properties can be applied to the MPF. The MPF, by definition, is a mnemonic technique due to its property of allowing the user to reconstruct the password for any given system by remembering only the MPF and having contextual knowledge of themselves and the system. Other mnemonic techniques can be applied to help remember the MPF itself. This second layer of mnemonics may also be tailored to the user of the MPF.

Given the authenticating user and the authenticating system, an adequately complex, long, and easy to remember MPF like the following could be constructed:

$$< u > @ < h|n > . < d|n >;$$

This MPF contains three elements: $< u >$ represents the first letter of the username, $< h|n >$ represents the first letter of the hostname *or* first number of the first address octet, and $< d|n >$ represents the last letter of the domain name suffix *or* last number of the last address octet. The modified MPF also contains a third special character in addition to the exclamation mark and period: the semicolon after the final element.

The above MPF would yield such passwords as:

- "d@n.v;" for user druid at system neo.jpl.nasa.gov
- "i@i.t;" for user intropy at system intropy.net
- "t@n.g;" for user thegnome at system nmrc.org
- "d@1.3;" for user druid at 10.0.0.33

Unlike the previously discussed MPFs, the one mentioned above employs a secondary mnemonic technique by reading in a natural way and is thus easier for a user to remember. The MPF can be read and remembered as "user at host dot domain," which is equatable to the structural format of an email address. Also, a secondary mnemonic technique specific to the user of this MPF was used by appending the literal semicolon character. This MPF was designed by a C programmer who would naturally remember to terminate her passwords with semicolons.

3.3.6 Advanced Elements

MPFs can be made even more complex through use of various advanced elements. Unlike simple elements which are meant to be replaced entirely by some static value like a username, first letter of a username, or some part of the hostname,

advanced elements such as repeating elements, variable elements, and rotating or incrementing elements can be used to vastly improve the MPF's output complexity. Note, however, that overuse of these types of elements may cause the MPF to not meet design goal number four by making the MPF too difficult for the user to remember.

Repeating Elements

MPFs may yield longer passwords by repeating simple elements. For example, an element such as the first letter of the hostname may be used twice:

$$< u > @ < h|n >< h|n > . < d >;$$

Such repeating elements are not required to be sequential, and therefore may be inserted at any point within the MPF.

Variable Elements

MPFs can yield more complex passwords by including variable elements. For example, the MPF designer can prepend the characters "p:" *or* "b:" to the beginning of the to include an element indicating whether the target system is a personal or business.

$$< p|b >:< u > @ < h|n > . < d|n >;$$

To further expand this example, consider a user who performs system administration work for multiple entities. In this case the variable element being prepended could be the first letter of the system's managing entity:

$$< x >:< u > @ < hi|n > . < d|n >;$$

$< x >$ could be replaced by "p" for a personal system, "E" for a system within Exxon-Mobil's management domain, or "A" for a system managed by the Austin Hackers Association. Most of the elements used thus far are relatively simple variable elements that derive their value from other known contextual information such as user or system name. The contrast is that elements are capricious only in how their value changes when the MPF is applied to different systems. Variable elements change values in relation to the context of the class of access or due to a number of other factors outside the basic "user/system" context.

To illustrate this concept, the use of the same MPF for a super-user and an unprivileged user account on the same system may result in passwords that only differ slightly. Including a variable element can help to mitigate this similarity. Prepending the characters "0:" or "1:" to the resultant password to indicate super-user versus

unprivileged user access. Respectively, by inclusion of an additional variable element in the MPF will result in the password's increased complexity as well as indicating class of access:

$$< 0|1 >:< u > @ < h|n > . < d|n >;$$

Variable elements are not required to prepend the beginning of the formula as with the examples above; they can be easily appended or inserted anywhere within the MPF.

Rotating and Incrementing Elements

Rotating and incrementing elements can be included to assist in managing password changes required to conform to password rotation policies. A rotating element is one which rotates through a predefined list of values such as "apple", "orange", "banana", etc. An incrementing element such as the one represented below by $< \# >$ is derived from an open-ended linear sequence of values incremented through such as "1", "2", "3" or "one", "two", "three". When a password rotation policy dictates that a password must be changed, rotate or increment the appropriate elements:

$$< u > @ < h|n > . < d|n >;< \# >$$

The above MPF results in passwords like "d@c.g:1", "d@c.g:2", "d@c.g:3", etc. To further illustrate this principle, consider the following MPF:

$$< u > @ < h|n > . < d|n >;< fruit >$$

The above MPF, when used with the predefined list of fruit values mentioned above, yields passwords like "d@c.g:apple", "d@c.g:orange", "d@c.g:banana", etc.

The only additional pieces of information that the user must remember other than the MPF itself is the predefined list of values in the rotating element, and the current value of the rotating or incrementing element.

In the case of rotating elements this list of values may potentially be written down for easy reference without compromising the security of the password itself. Lists may further be obscured by utilizing certain values, like a grocery list or a list of company employees and telephone extensions that may already be posted within the user's environment. In the case of incrementing elements, knowledge of the current value should be all that is required to determine the next value.

3.4 Enterprise Considerations

Large organizations could use MPFs assigned to specific users to facilitate dual-access to a user's accounts across the enterprise. If the enterprise's Security Op-

erations group assigns unique MPFs to it's users, Security Officers would then be able to access the user's accounts without intrusively modifying the user's account or password. This type of management could be used for account access when user is absent or indisposed, shared account access among multiple staff members or within an operational group, or even surveillance of a suspected user by the Security Operations group.

3.5 Weaknesses

3.5.1 The "Skeleton Key" Effect

The most significant weakness of passwords generated by MPFs is that when the formula becomes compromised, all passwords to systems for which the user is using the respective MPF schema are potentially compromised. This situation is no worse than a user simply using the same password on all systems. In fact, it is significantly better due to the resultant passwords being individually unique. When using a password generated by an MPF, the password should be unique per system and ideally appear to be a random string of characters. In order to compromise the formula, an attacker would likely have to crack a significant number of system's passwords which were generated by the formula before being able to identify the correlation between them.

3.5.2 Complexity Through Password Policy

A second weakness of MPF generated passwords is that without rotating or incrementing elements, they are not very resilient to password expiration or rotation policies. There exists a trade-off between increased password security via expiring passwords and MPF complexity. However, the trade-off is either to have both, or neither. The more secure option is to use both, however, this practice increases the complexity of the MPF potentially causing the it to not meet design goal number four.

Chapter 4

Conclusion

MPFs can effectively mitigate many of the existing risks of complex password selection and management by users. However, their complexity and mnemonic properties must be managed very carefully in order to achieve a comfortable level of password security while also maintaining memorability. Users may reintroduce many of the problems MPFs intend to solve when they become too complex for users to easily remember.

Bibliography

[1] Bugaj, Stephan Vladimir. *More Secure Mnemonic-Passwords: User-Friendly Passwords for Real Humans"*
http://www.cs.uno.edu/Resources/FAQ/faq4.html

[2] Kotadia, Munir. *Microsoft Security Guru: Jot Down Your Passwords*
http://news.com.com/Microsoft+security+guru+Jot+down+your+passwords/2100-7355_3-5716590.html

[3] McWilliams, Brian. *How Paris Got Hacked?*
http://www.macdevcenter.com/pub/a/mac/2005/01/01/paris.html

[4] Williams, Randall T. *The Passphrase FAQ*
http://www.iusmentis.com/security/passphrasefaq/

[5] Jeff Jianxin Yan and Alan F. Blackwell and Ross J. Anderson and Alasdair Grant. *Password Memorability and Security: Empirical Results*
http://doi.ieeecomputersociety.org/10.1109/MSP.2004.81