Secure Coding.
Practical steps to defend your web apps.

SFSP Tutorial

Simple Formula for Strong Passwords:

Dramatically Increase Information Security with Minimal Training, and Without Costly Infrastructure Changes

GSEC Practical Assignment 1
Version 1.4c
Option 1 – Security Research

Bernie Thomas
March 19, 2005

Abstract

The practice of using passwords for user authentication exposes organizations' and individual users' data to disclosure alteration and/or destruction. However, a large portion of the security issues that make this true can be satisfactorily addressed using a simple method that I would like to introduce as the Simple Formula for Strong Passwords (SFSP) [Note 1]. This manual method creates very strong passwords averaging 10 – 14 characters, with upper and lower case letters, special characters and numbers. It does so using a *memory friendly* method that removes the need and the desire to improperly store passwords (such as under the keyboard, behind the calendar, on a sticky note, etc.). It also allows users to easily remember and adjust to mandatory password changes and requirements that different applications have unique passwords. With SFSP tendencies to create simple/weak passwords like *john* are transformed with an easily remembered and repeatable technique to create strong passwords like 1;j718Ohn; or ^j7O1h8N^. SFSP can be used to create new strong passwords or to strengthen old familiar ones. The most valuable feature of SFSP is that is allows users to accurately recall strong passwords from memory, even after long periods of non-use.

Note
1) To the best of my knowledge this concept is original. I have been unable to find other works describing methods similar to it. Until I do I will refer to it as Simple Formula for Strong Passwords – SFSP

Problem

Passwords are plagued with problems and vulnerabilities that make them inadequate for many of the authentication roles for which they are used. Consequentially passwords are falling out of favor with many security professionals (Hensing). Security professionals are pushing for multi-factor authentication, also known as strong authentication, in order to compensate for password shortcomings. However, cost and complexity issues have slowed multi-factor authentication's adoption. Consequently most of us are still left to make the best use of what we currently have. Passwords.

What makes passwords so bad? They tend to be easily compromised (Meaning that they can be discovered). Why are they easy to compromise? Because the use of Weak Passwords, Null Passwords, Default Passwords, Improper Password Storage, Clear-Text Passwords (A.K.A. Plain Text) (SANS InfoSec Reading Room, Mortenson) and Insufficient IT Controls (SANS Top Twenty) allows successful inference (guessing), cracking and pre-computational attacks on passwords.

Guessable Passwords (inference vulnerability) are password choices that are obvious to or determinable by other people (these are usually names and/or dictionary words). These are easy for users to remember, but because of that, they are also easy for unauthorized users to determine.

Some examples of guessable password categories are:

Type	Percent of Users	Description
1. Family Names	50%	• Names, including Pets
2. User's Favorites	30%	• Sports Team, Singer, Actor, etc
3. Self Descriptors	11%	• Such as "stud" or goddess
4. Cryptics	9%	• Users of strong passwords

Illustration A (McAuliffe)

Cracking Passwords is a phrase that describes being able to run programs against hashed (encrypted) passwords in order to reveal the underlying password. Generally speaking, the longer the password the harder it is to crack. The truth is that length and complexity (using all possible characters – letters, number, and special characters) are both needed to create the most resistance to this type of attack (Shaffer).

Pre-computation Attacks describes the practice of pre-computing the password hashes* (encrypted state of a password) for all possible character combinations. This creates a list that is then compared against legitimate password hashes (Hensing[2]). When matching hashes are discovered the underlying password is compromised. This is very effective except when password strength makes computing and storing the hashes too time consuming or difficult. Currently Strong Passwords can defeat this technique with as few as 8 characters (Strong Passwords are defined on page 5).

* *Password Hashes* are the encrypted representation of a password. Weak passwords create hashes that can be compromised when captured (Cracking and Pre-computation). This is why sniffers are password threats even when password encryption is being used (*Sniffers* are devices that allow computer communications to be monitored and recorded).

Weak Passwords are passwords that are guessable, crackable or vulnerable to pre-computational attacks.

Null Passwords is a phrase that describes when no password has been assigned to an account. The password field is blank, or "Null". This represents a big convenience for users, but opens the door for easy unauthorized access.

Default Passwords are passwords that vendors include in their products *as delivered* from the company. Leaving default passwords makes them easier to remember or recover when forgotten, but also means that everyone else that knows or learns that default password may be able to compromise your system. The internet is full of default password lists for those that want to know the default password to your particular system.

Improper Password Storage is when passwords are stored in fashions that allow the passwords to be discovered. The most egregious form of this is when users write passwords on sticky notes and put them on their monitor, under their keyboard or behind their desk calendar (and other similar practices). It is also improper password storage for a user to type all of their passwords in a document or text file and save it unencrypted where others can access it (It's even worse to name the file "Passwords").

Clear Text Passwords are passwords that are sent in plain readable text, unencrypted. They allow people watching the network to easily see the password and login credentials (Mortensen).

Insufficient IT Controls describes an IT environment where common protections are not in place. Common protection are things like disabling all LM (LAN Manager) Authentication across the network, preventing LM hashes from being stored, maintaining strong password policy, enforcing strong password policy, firewalls, intrusion detection systems, intrusion prevention systems, etc. (SANS Top Twenty).

Solution

Generally solutions for these problems are:

Common Problem	Common Solution
1. Guessable Passwords	• Strong Passwords
2. Cracking Passwords	• Strong Passwords
3. Pre-computational Attacks	• Strong Passwords
4. Weak Passwords	• Strong Passwords
5. Null Passwords	• Strong Passwords
6. Default Passwords	• Strong Passwords
7. Improper Password Storage - Writing Down Passwords	• No viable solution is in wide use. Many unsuccessfully attempt forcing proper storage with policies. A few suggest storing passwords in secure cabinets/safes. Not practical for everyday, multi-instance use
8. Clear Text/Plain Text Passwords	• Don't send plain text. Encrypt tunnels or upgrade to applications that support encryption
9. Weak Password Hashes	• Strong Passwords - makes hash marks too large and complex for rainbow-like tables to be able to calculate them and takes too long to crack (lifetimes)

Illustration B

Notice that most of the problems in Illustration B would be solved with consistent Strong Password use. In fact, *all* of the strictly password related problems here are solved with *Strong Passwords* except for Improper Password Storage. (SFSP provides for Proper Password Storage *and* Strong Passwords).

What is a Strong Password?

I generally think of a good password as one that takes longer than your password change interval to crack. The term *Strong Password* however is beginning to take on the specific meaning of one that uses all possible password strengthening options; such as use of multiple upper AND multiple lower case letters, use of numbers, use of multiple special characters and use of at least 8 total characters.

Webopedia suggests that a strong password is one that is not easily determined by humans or machines (Internet.com). Microsoft also associates the difficulty of password determination as the result of a Strong Password, "The intent of a strong password is to make the password harder to guess, and thus more secure" (Microsoft, Security Quiz). Both Microsoft and Webopedia also claim that a larger character set is better at creating Strong Passwords. However, these definitions disagree on how many characters are best. This is not uncommon. System administrators know that the more characters are required in a password, the more likely it will be forgotten or written down. Also, some administrators are more sensitive to exploits and want longer passwords for added security, while other administrators are more sensitive to users' capabilities and want shorter passwords to reduce the chances of improper password storage.

Consequently some security policies overtly state that the definition of Strong Passwords will be left up to the individual implementation of the password. For example, The University of Illinois policy states, "All users of systems that contain high risk or confidential data must have a strong password - the definition of which will be established and documented by UTMT <University Technology Management Team>after consultation with the community. Empowered accounts, such as administrator, root or supervisor accounts, must be changed frequently, consistent with guidelines established by UTMT" (Business and Financial Pol.). Picking the right password and password policy has become so difficult that extensive information has to be provided to users when password policies are valued and enforced by management. For examples of this type of help see:

- "Choosing a Good Password" http://www.cs.umn.edu/help/security/password-selection.php (University of Minnesota)
- "Tips for Choosing a Password" http://www1.umn.edu/oit/security/passwordguide.html (University of Minnesota)

There is a direct conflict between the solution of Strong Passwords in Illustration B, and the user response to Strong Passwords. That response is to improperly store passwords. The stronger the password requirements that are put on users, the more likely they are to improperly store their passwords. This is simply because stronger passwords are more difficult to remember.

There is therefore an inverse relationship between password strength and proper password storage. This being the case, to get users to stop the writing down of passwords would require allowing passwords that are easy to remember. Then the user will not need to write them down. This solution is unacceptable because it makes passwords more easily guessed and cracked. The typical policy response to this information is to require that passwords are longer, more random, use more characters and be enforced through technology. But then the cycle starts again as this makes passwords difficult to remember. So users write them down again. It is a cycle where solving one problem creates another problem. This leads to the conclusion, which many security professionals have reached already, that passwords are not a feasible means of providing dependable authentication.

None of this would be the case if users could easily remember 10 – 14 character passwords that were derived from a 92 character set and consisting of a mix of all the types of available characters– letters, numbers, special characters, special keys (which, with current technology could take thousands of years to break (Shaffer)). This is to say that if we could sufficiently increase the memory capability of every user to be able to intuitively remember strong passwords, then passwords would be a good means of providing reliable authentication.

If we could then incent the users to remember and want to guard their passwords we would have a situation where passwords were properly stored and inherently too strong for cracking. SFSP does just that. Let me explain. Go back to our list of password problems from Illustration B. If we reproduce it, adding SFSP in where it is appropriate you will see that nearly all password vulnerabilities are addressed. That's because most of the vulnerabilities are related to weak passwords and improper storage (see Illustration C - next page).

Common Problem	Common Solution
1. Guessable Passwords	• Strong Passwords through SFSP
2. Cracking Passwords	• Strong Passwords through SFSP
3. Pre-computational Attacks	• Strong Passwords through SFSP
4. Weak Passwords	• Strong Passwords through SFSP
5. Null Passwords	• Strong Passwords through SFSP
6. Default Passwords	• Strong Passwords through SFSP
7. Improper Password Storage - Writing Down Passwords	• Easy to remember Passwords (through SFSP)
8. Clear Text/Plain Text Passwords	• Don't send plain text. Encrypt tunnels or upgrade to applications that support encryption
9. Weak Password Hashes	• Strong Passwords through SFSP - makes hash marks too large and complex for rainbow-like tables to be able to calculate them and takes too long to crack (lifetimes)

Illustration C

Notice how simply using and properly safeguarding (storing) Strong Passwords can address so many of the password problems we have discussed. The remaining vulnerabilities in Illustration C deal with people covertly watching the I/O (Input/Output) of a computer. Part of that is addressed with SFSP. For example, whenever passwords or virtual passwords are sent from a computer encrypted, if SFSP has been used, then the resulting hashes will be too difficult for crackers to be able to make use of. However, what is not addressed by SFSP is the sending of passwords over networks in Plain Text, and the monitoring of computer Input/Output (I/O) by hardware/software devices such as keystroke loggers. These are not the main focus of this paper because they are not vulnerabilities limited to passwords. However, I will touch on them in this paper when I will demonstrate how using a few simple techniques can safeguard users from all but the most sophisticated password grabbing attacks. That includes controls to mitigate risks resulting from weak passwords, improper storage of passwords, network sniffing *and* keystroke logging.

SFSP

So what is the answer? SFSP. What is SFSP? SFSP is part of an umbrella strategy that I have come up with that stresses simplicity and common sense thinking for developing low cost technology alternatives for smaller cash-strapped companies/departments.

SFSP is not intended to replace common IT controls for protecting against password vulnerabilities. It is meant to work in harmony with those common practices. Organizations still need policies that enforce password history, minimum and maximum password ages, strong passwords, non-reversible encryption for stored passwords, and the disabling of Windows LAN Manager and LM Hashing (SANS Top Twenty) .

Having said that, and to overly simplify, SFSP is a way to easily remember strong passwords. It accomplishes this goal by providing a mechanism for deriving strong passwords across all password uses, and like a password, the semantics of its mechanisms are only known to the individual user. Picture this; tell a user to always use the name of a site or application as their password and you would have a mechanism for creating unique passwords, though not a very good one. In this case a user providing a password to login to a UnaTime timesheet would use “UnaTime” as their password. They would use “hotmail” as their password to login to their Hotmail email site and “Word” to access password protected Word documents. These illustrate using a consistent mechanism to create unique passwords. They remain unique as long as each user is given a different mechanism by which to create their passwords. However, the passwords in these examples are neither secret nor strong, and most likely are not unique either.

SFSP provides a mechanism for consistent password generation that results in secret, unique, strong passwords. Additionally, with SFSP users will easily be able to reproduce them even after long periods of not using the password, and they will be able to compensate for password changes that are mandated by maximum password age policies. I will illustrate how SFSP works, but will stick to the most simple examples first. Keep in mind that general cipher principals can be applied to SFSP to make it even stronger. That type of advanced implementation is just as easy for users. I will include some examples of Advanced SFSP implementations later on. Illustration D shows the basic makeup and components of SFSP.

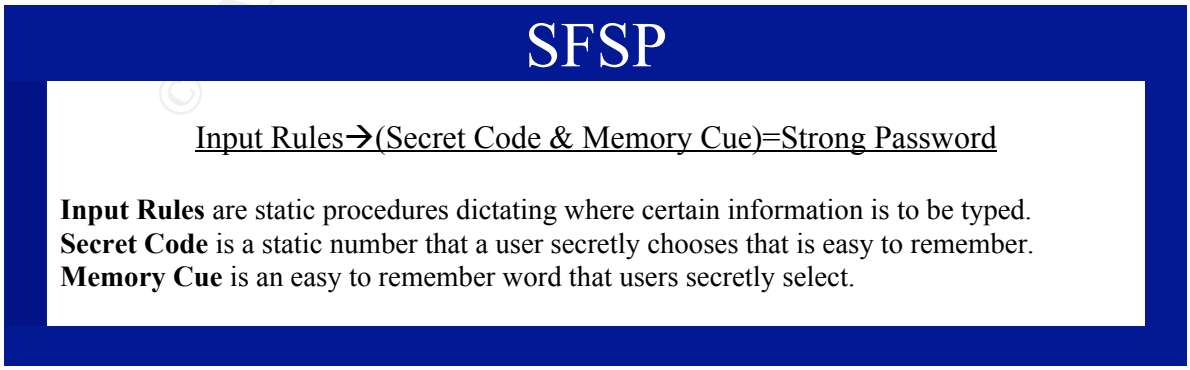


Illustration D

Input Rules

Input Rules form an input mask that dictates what information goes where in a password. For example, to decide that the second character of my password must always be a capitalized letter is to set an input rule. Any password subject to this Input Rule would be modified as demonstrated in Illustration E:

Original Password	New Password
system	sYstem
molasses	mOlasses
Administrator	ADministrator

Illustration E

Input Rules never change because we want users to remember them. We also want users to eventually start creating more advanced input rules for themselves and, as part of defense-in-depth to keep those secret. More advanced input rules create passwords that are difficult to decipher even in plain text. For example, a simple rule of alternating descending numbers before, after and through the characters of a simple word would create the password **5J4o3h2n1** from the simple word "John". 5J4o3h2n1 by itself would be considered a relatively good password by most, and it only uses one of the three components of SFSP. It would not, even in plain text be obvious to most eyes that it is simply the word "John" mixed in alternating fashion with numbers.

Changing Input rules further by making the numbers decrease by two adds even more complexity for prying eyes, but not to the users: **8J6o4h2n1**. Users very quickly get used to the rote typing dictated by the input rules, but casual observers will find it more difficult to recognize, guess or crack the passwords. Consider this set of Input Rules with some of the original passwords from Illustration E.

Input Rule Example A:

Rule Number	Rule Description
R1	• Add doubling numbers in between each character of a simple word, before, through and after.

Results in:

Original Password	New Password
system	1s2y4s8t16e32m64
molasses	1m2o4l8a16s32s64e128s256

Illustration F

This illustrates how much an Input Rule can both obscure a simple password's root and create a stronger password. These Input Rule examples are too simple to recommend for use in a production environment. However, the more substantial rule sets are just as easy for users to remember.

I refer to Complex Input Rules as Strong Rules because they result in Strong Passwords. Strong Input Rules make it possible to include the largest possible character set in easy to remember passwords. Consider the previous Input Rule example as we add another rule to make our password a stronger password:

Input Rule Example B:

Rule Number	Rule Description
R1	• Add doubling numbers in between each character of simple word, before, through and after
R2	• Add the special character "*" (not including quotes) to the very beginning and very end of the password

Results in:

Original Password	New Password
system	*1s2y4s8t16e32m64*
molasses	*1m2o4l8a16s32s64e128s256*

Illustration G

Or

Input Rule Example C:

Rule Number	Rule Description
R1	• Add doubling numbers in between each character of simple word, before, through and after
R2	• Insert the special character "*" (not including quotes) after the second and last character, as the last step in creating the password

Results in:

Original Password	New Password
system	1s*2y4s8t16e32m64*
molasses	1m*2o4l8a16s32s64e128s256*

Illustration H

VARIABLE CHARCTERS

Another function of Strong Input Rules is that they allow users to compensate for mandatory password changes as well as unexpected password resets/account lockouts. SFSP handles these events through variable password characters. As you will see through this discussion, much of what goes into an SFSP password is static. It is this lack of change that makes SFSP passwords easy to remember. That is discussed later. However, one module of SFSP Input Rules addresses a variable component of SFSP passwords. It is this variable component that compensates for change.

Typically I recommend that company passwords have date fields as their variable component and that internet passwords use what I call a sequence character as its variable piece. Let's start with Internet Sequence Characters. Look back at the first password example made with Input Rules (Illustration E) and you will see the following:

Original Password	New Password
system	sYstem

Illustration I

Variable Sequence Characters - *Password Resets/Account Lockouts*

Remembering that our first mention of a simple Input Rule was: *Capitalize the second character of every password*. Consider this; in the event that a user makes enough data entry mistakes while logging into an account that they are locked out and forced to reset their password to something new. This is especially common with financial (Banking, Credit Cards, etc) websites. In order to keep password management simple a user could have an input rule that mandates that every time a password is changed against the user's will, the same password will be used again and a sequential counter will be at the beginning of the password. If this method were being used, each time the password was forcibly reset a sequence character would be modified upward. It would look like this:

Rule Number	Rule Description
R1	• Capitalize the second character of the password
R2	• Lastly add a SEQUENCE character (1,2,3/a,b,c) in front of the password (one Variable Character)
R3	• Lockout Rule - Increase the sequence character by one at every forced password reset

Results in:

Original Password	New Password
system	1sYstem

	New Password
Password after forced reset #1	2sYstem
Password after forced reset #2	3sYstem
Password after forced reset #3	4sYstem

Illustration J

The sequence character is important because it helps the user manage passwords more easily. It does this by making the guesses after a failed login attempt predictable. Picture this, a user checks their web-based email once per month. Their password is "1sYstem" according to the sample rules in Illustration J. After not using the account for a month the user attempts to log in and fails with a credentials error.

The user has forgotten that they were locked out last month and forced to enter a new password. However, when they entered the new password they followed their Input Rules and created the password "2sYstem".

Because the user has forgotten about the lockout they try their original password, "1sYstem" again and receive the same failure message. Because they were trained on SFSP they realize that because of their input rules, there are only a few possibilities for what the password can be. After trying their first password, "1sYstem" twice, they are sure it is not working. What is their next logical password to attempt? Naturally, if they are using SFSP the next password the will try is "2sYstem".

> Note: When the user tries increasing the sequence part of the password they will eventually find the correct password. This is assuming they allow sufficient time in between attempts to avoid another locked account, which they should be

> trained to do. When training on the SFSP method users need to be educated about how to avoid and recover from lockouts. Train users to be very careful when logging in so as to avoid typing mistakes. After a failed login message is received, they need to be confident of what they typed so they can try something else (instead of wasting another attempt on the same password). Also train them to be sure of what they type as a second attempt, and to remember it so that their third attempt is certain to be different yet again. However, after a second failure ask users to wait for a period before attempting a third time. This is because a third failure often results in an account lockout. Time to wait before third attempt varies by use-case. Training should be specific to users' environments.

The sequence/lockout character is good for internet use because it allows users to figure out passwords for accounts that have had their passwords changed more often than they wanted them to be (by forced resets and such). It is the simplest way of dealing with multiple, infrequent use accounts (whose passwords have higher probabilities of being forgotten). This is because SFSP users can find correct passwords by trying *the few possible* combinations that only they know could be correct. (My recommendation is that sequence characters are only one digit. Adding more complexity to the variable characters for internet use would defeat the purpose of SFSP because it would probably necessitate the writing down of passwords, or the use of password management software).

Variable Date Characters - *Mandatory Password Changes*

As discussed above, password management on the internet becomes too cumbersome if complex variable characters are used. However, in an office environment it is easy to create stronger passwords through greater complexity in the variable character fields. In an office environment we have the advantage of predictability and account unlocks; predictability in the sense that we know how often passwords are to be changed; and account unlocks in that we know that our IT department can unlock our accounts without forcing us to change their passwords. This means that in an office environment we can take advantage of constants and with the same or greater simplicity can add strength to our passwords through variable character complexity.

Here is how to do it. Use the month and year, in numerical form as the variable characters. This is very simple. For example, if you change your password in January 2005 then your variable character field could look like 012005 (from 01/2005). If it were created in February then it could be 022005 (from 02/2005). Here are some variable date character examples using the same base information that was used in the section above on sequence characters:

Password
system

Rule Number	Rule Description
R1	• Capitalize the second character of the password
R2	• Lastly add a DATE character in front of the password (multiple Variable Characters)

(Sequence rules are not usually needed in company network environments, but if one were to be added a third rule might look like this:

Rule 3: **Lockout Rule** – In the event of a forced password change before the anticipated password change interval, add a sequence character in front of everything. Increase that by one at every time this happens.

This would result in a password like 1012005sYstem).

Results in:

Date	New Password
1. January 2005	012005sYstem
2. February 2005	022005sYstem
3. September 2005	092005sYstem
4. January 2006	012006sYstem
5. February 2006	022006sYstem
6. September 2006	012006sYstem

Illustration K

To sum it up, as mentioned in the previous section, I recommend that sequence characters are implemented for internet passwords, and date fields for work related passwords. They both represent the variable parts of SFSP passwords that are needed to compensate for change while keeping the familiarity of existing passwords intact. Because internet sites rarely enforce regular password changes, the sheer infrequency and irregularity of previous password changes makes password change management more difficult than with office environments. To compensate for that, sequence characters allow users to simply try passwords sequenced by one character at a time. This is very easy to remember and not dependant upon dates. That makes sequence numbers ideal for internet passwords (though internet passwords should be stronger than the examples shown here thus far).

On the other hand, company passwords are changed regularly, every month, every quarter, every six months, for example. The company policy therefore indirectly dictates what the date field will be in a password. This predictability means that SFSP users can always know that their password is either this period's password or last period's password, and the difference between the two is small and easy to remember. This is not to say that there will never be a need for a sequence or character on company systems, or a date character in an internet system. But usually at work an account can be unlocked without having to be reset. This being the case, date characters are normally sufficient and easier to remember in the controlled environment of a work environment. Date characters also consist of more digits and therefore result in stronger passwords. However, in the event of a forced password reset a sequence character can be still be added in addition to date characters and then left off at the next normal password change.

Remember that Input Rules are a set of procedures used for mixing simple words (or Memory Cues) with Secret Codes to form stronger passwords, and which also instruct users to add set Special Characters and Variable characters to create truly strong passwords. The mixing method, simple or root (Memory Cue) word and special characters are the fixed components of the password and the Sequence or Date characters represent the variable component.

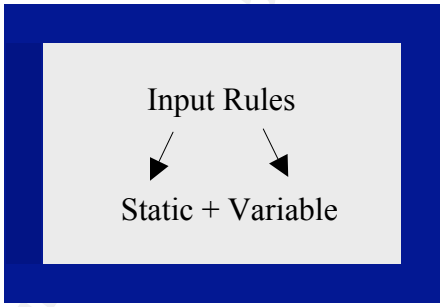


Illustration L

I will provide more examples of Strong Input Rules and Passwords after introducing both the Secret Code and the Memory Cue concepts. Until then I will use simple Input Rules.

> Note: Using Sequence based variable characters was designed specifically for internet use, whereas Date based variables were designed specifically for LAN use. Using Date based variable characters is referred to as Full or Regular SFSP, and using Sequenced based is referred to as Internet SFSP (iSFSP). Full SFSP is recommended to protect any critical resource (LAN or internet), and Internet SFSP is recommended for low to moderate protection needs.
>
> The illustration below provides a quick reference guide for when to use Sequence versus Date based characters.

Recommendations for Password Use

Use Type	Method to Use
1. Office	• Full SFSP
2. Home or Home Office	• Full SFSP
3. Internet – Critical	• Full SFSP
4. Internet – Typical Use	• SFSP with sequence characters (iSFSP), or coordinate these password changes with other critical password needs and use Full SFSP
5. Internet – Disposable Logins	• Disposable Password – One easy to type and remember password for use-cases not requiring strong password (like web conference registrations and other one-use situations). This common practice is acceptable but not part of SFSP and therefore not covered in this document

Illustration M

Secret Codes

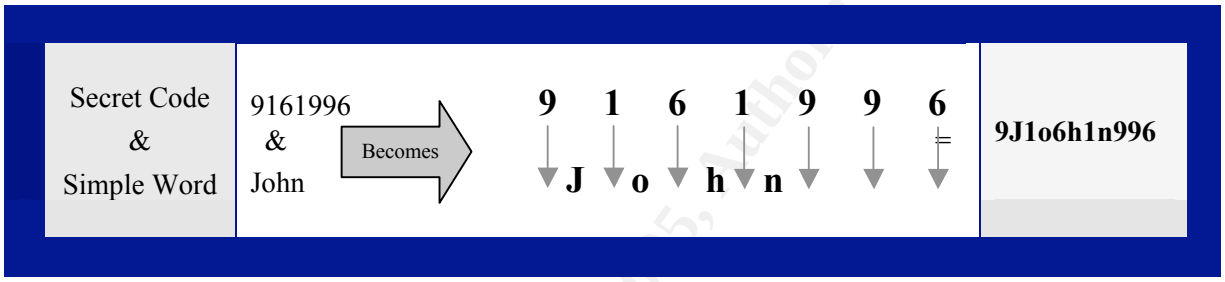
The Secret code is a number that a user picks that he or she can easily remember. It always stays the same, and is in every password that a user creates. That makes it easy to remember. Contrary to typical password scenarios, Secret Codes can be a birthday, wedding date or something else guessable. This is why it is easy to remember. The fact that it will be mixed with other elements to form a password makes the final password non-guessable, even if the Secret Code is known.

The previous section discussed Input Rules. In that discussion I arbitrarily picked numbers and mixed them with a simple word in order to create a better password. With SFSP it is the Secret Code that is mixed with a simple word in order to create the better password. The first two samples of passwords were **5J4o3h2n1** and **8J6o4h2n1**.They represented the simple word “John” mixed with 5 alternating and descending numbers. (Descending order numbers tend to be more difficult for users to get accustomed to, but were good at demonstrating points made in that section. In reality, to keep things simple I would not ask users to pick descending numbers).

Let’s look at the same simple word mixed with a Secret Code rather than with arbitrary number selections. Consider this example; Shirley is creating a new password to login to her computer. In the past she has stuck with using either “John”, her husband’s name, or a numerical representation of her wedding date 9161996 (September 19, 1996). Since her IT department is starting to enforce some password rules, 8 – 14 characters with letters AND numbers, she thinks that she must give up her easy to remember passwords. The easiest solution for Shirley would be to simply combine the two passwords. This would represent a simple word being mixed with a code number to form a better password. In essence this is what SFSP will do for Shirley, only it will not do it is such a way that the two combined words create another single word that is easy

to discern. In this example Shirley's wedding date represents her Secret Code and "John" is the simple word that she combines it with. Then her password becomes "John9161996" (or inversely, 9161996John) and meets the IT department's requirements. It also makes a stronger password that is easy to remember. She is not likely to write this one down.

If we again add in the concept of defense-in-depth we realize that though her password is better and meets the IT department's requirements, it is also easily recognizable to sniffers and shoulder surfers. It is also guessable. To make the password even more secure and to fool the naked eye, Shirley mimics the format of the first password examples I described (**5J4o3h2n1** and **8J6o4h2n1**), and uses her Secret Code (9161996 from September 19, 1996) instead of the arbitrary numbers that I did (which were 5,4,3,2,1 and 8,6,4,2,1). Then her password becomes **9J1o6h1n996**.



Notice that since the simple word "John" was not long enough, we simply finished writing out the Secret Code after "n". This may be a good time to add that the Input Rules are customizable procedures for building a password. It is not necessary though to pick password components (i.e. Input Rules, Secret Code and Simple Password Root) that perfectly match in length. One might want to pick a four or five digit Secret Code to go with a simple word like "John", and produce a password like 1J2o3h4n5, but it is not necessary. Shorter and longer Secret Codes – 1J2o3hn/1J2o3h4n5678 - are fine as long as the final result is a sufficiently long password (10 – 14 characters).

Illustration N

Back to the example - Shirley's password is now **9J1o6h1n996**, and it is much stronger than anything she has used in the past. It is not guessable, nor is it easily discernable when seen, and while few people could remember it once seeing it, it is easy for her to remember. The components of it are burned into her memory, and the only thing she had to memorize was the method of blending what she already knew. (Don't forget, because of things like rainbow tables, passwords should contain a larger character set than what Shirley is using here (Kahn). Her password only represents a set of 62 possible characters – 52 upper and lower case letters and 10 numerical digits. A set of 94 possible characters is recommended, and use of special keystrokes is also encouraged).

So in this case the Secret Code is Shirley's wedding date. It could have been anything she wanted to combine with the simple word that she picked ("John"). See the illustration below for more of Shirley's possible Secret Codes.

Secret Code Example 1- mixed with previous Input Rule principals):

Simple Word
John

Secret Code Source	Secret Code Number	New Password
1. Her wedding date	• 9161996 (Sept. 16, 1996)	• 9J1o6h1n996
2. Her favorite number	• 777	• 7J7o7hn
3. Her child's birthday	• 6011998 (June 6, 1998)	• 6j0o1h1n998
4. Her apartment number	• 2516	• 2J5o1h6n
5. The ages of her family members from youngest up	• 36307 (36,30,7)	• 3J6o3h0n7

Secret Code Implementation Example 2 - Using previous password list as simple words and Shirley's wedding date (Sept. 16, 1996) as the Secret Code:

Original Password	New Password
1. system	9s1y6s1t9e9m6
2. molasses	9m1o6l1a9s9s6es
3. Administrator	A9D1m6i1n9i9s6trator

Illustration O

Memory Cue

The Memory Cue is simply an easily remember password root, and the final component of the SFSP. It is a word that a user picks that they are sure to remember. That word is then included in every password they create. How it is included depends on the Input Rules described earlier. The Memory Cue is by nature supposed to be simple, like the rest of SFSP. That is what makes it easy to remember. In the section above Shirley used a simple word (John) and mixed it with her static Secret Code in order to strengthen her password. That simple word was only good for one unique password though. If it becomes necessary for Shirley to have multiple unique passwords in her network environment she should pick simple words that are different, and which will relate to what she is doing when prompted for a password. This Relative Simple Word concept is referred to as the Memory Cue. It will cue Shirley as to what her password root should be for any given use case.

Let's look back at Shirley's example. Shirley had the following components:

Secret Code	Simple Word/Memory Cue
9161996	John

I previously gave examples of Input Rule sets that could be used, but had not assigned one to Shirley. Let's assume that these are her only Input rules:

Rule Number	Rule Description
R1	• Insert Secret Code after third character of simple word/Memory Cue
R2	• Only capitalize the first character of the simple word and the first character after the Secret Code

Illustration P

Using Illustration (D) with which we introduced SFSP, Shirley's equation would look like this:

Shirley's SFSP

Input Rules→(Secret Code & Memory Cue)=Strong Password

R1, R2 →(9161996 & John) = Joh9161996N

Shirley's Stronger Password = Joh9161996N

Note: The two main attractions of SFSP are that users remember passwords without improper password storage (writing them down), and that the users have very strong passwords. It should also be noted that a hacker could have any two of the components of SFSP and still not be able to guess the resulting password (though the more components a hacker has compromised the more likely that brute force cracking programs could be customized to gain the password).

Illustration Q

This resulting password is fairly good by most lay standards and will be incredibly simple for Shirley to remember, but no one else. Only she knows that she takes her

husband's name and the date they married and mixes them every-other-character to form her password (that mixing method represents her input rules).

As described above Shirley's Memory Cue of "John" is not really appropriate for an environment where many different passwords are required, especially where they are dependant upon the application, file, website, etc. that is being used. She may prefer to use the same password for everything, but in accordance with defense-in-depth practices it is recommended that policies require different passwords for different uses (as well as a different set of passwords for internal versus external access).

To help users like Shirley create multiple unique passwords that are easy to remember over time she will use a Memory Cue, something to remind her of what password she would have or should have created for a particular use-case. For example, assume that Shirley password protects all of her documents. She may choose the name of the document to be her Memory Cue. If her document is named BUDGET, then in this case the word "BUDGET" would be her Memory Cue. Taking this into consideration with what is already established about Shirley, she has enough information to make her password for the BUDGET document, as is illustrated below:

Secret Code	Memory Cue
9161996	BUDGET

Rule Number	Rule Description
R1	• Insert Secret Code after third character of Memory Cue
R2	• Only capitalize the first character of the Memory Cue and the first character after the Secret Code

Shirley's SFSP

Input Rules→(Secret Code & Memory Cue) = Strong Password

R1,R2→ (9161996 & BUDGET) = *Bud9161996Get*

Shirley's Stronger Password = *Bud9161996Get*

Illustration R

Using this method Shirley will always be able to remember the password for any document that she has password protected. Instead of remembering the written form of her formula (Illustration Q), she will simply know to insert her Secret Code (which never changes) after the third character of the document name that she is opening (Users find that putting Input Rules into plain English like this is *very* helpful).

This is the concept of the Memory Cue. In practice this exact implementation of Memory Cues may create excessively long passwords, as document names can be very long. In most real world applications my suggestion is to always use only part of the name of a Memory Cue in password creation. I call this the Truncation Rule. For example only use the first or last five characters of a Memory Cue. Had Shirley done that in the previous example her Memory Cue of BUDGET would have been truncated to either BUDGE or UDGET, and the resulting password would have been one of the following:

First five characters: *Bud9161996Ge*
Last five characters: *Udg9161996Et*

In the event of a long (Memory Cue) document name, the method works as easily. Assume Shirley wants to password protect the document named, "**Workforce Planning Proposal – Tim**". Using Shirley's Input Rules and Secret Code, and the Truncation Rule, we have the following possible new passwords:

Memory Cue Truncation

Original Memory Cue	Truncated with
Workforce Planning Proposal – **Tim**	First 5 Characters = Workf Last 5 Characters = - Tim **(Notice that I included the space from the last 5 characters of the Memory Cue. This is a password strength enhancement. But some applications will not allow spaces in passwords. In that case Input Rules should expressly ignore spaces. If Shirley's input rules ignored spaces her Last Five Truncated Memory Cue would be "L-TIM" instead of "- TIM")**

Shirley's corresponding passwords would be:

Password with FIRST Five Truncation	Password with LAST Five Truncation
Wor9161996Kf	t9161996Im

Notice that Shirley's rule on capitalization cannot affect special characters and spaces so it is ignored in those instances, and yes, passwords can start with spaces though this one does not.

Illustration S

At this point Shirley has decided to only use the first five characters of her Memory Cues. She understands that if she has a Memory Cue with less than five characters, like SAM, it's no problem. She simply carries out the actions that she can with the characters she has.

The Memory Cue concept works the same way for any password use-case. Let's review what Shirley's SFSP components are and see what passwords would be produced for various use-cases:

Memory Cue Example 1- mixed with previous Input Rule principals):

Secret Code
9161996

Rule Number	Rule Description
R1	• Insert Secret Code after third character of Memory Cue
R2	• Only capitalize the first character of the Memory Cue and the first character after the Secret Code
NEW ▶ R3	• Truncate using only the first five letters of Memory Cue

Illustration T

Use Case	Resulting SFSP Password
1. Document named, BUDGET	• Bud9161996Ge
2. Spreadsheet named, Expense List	• Exp9161996En
3. Windows Login (Cue = Windows)	• Win9161996Do
4. Hotmail Account	• Hot9161996Ma
5. Yahoo Account	• Yah9161996Ho
6. Unatime Web Timesheet	• Una9161996Ti
7. Intranet Account	• Int9161996An

Illustration U

To the individual user there is an obvious tie between the password use-case they are involved with and the actual password that they should be using. The word that describes that tie for that individual is the Memory Cue. The Memory Cue is simply an easy to remember name of the password use-case. If the password use-case is to log into a website called Hotmail, then the individual's Memory Cue could be HOTMAIL (or EMAIL or WEBMAIL, etc.). If the use-case is to open a document called RESUME, then the Memory Cue could be RESUME. Naturally a user can choose any Memory Cue for any given password use-case. In my example the Memory Cue for logging into Hotmail is simply HOTMAIL. But another user may only use their Hotmail account to communicate with their mother, and therefore want to use the Memory Cue of MOM for that site. All the components of SFSP are subject to what the user wants them to be. Though, it may be helpful when first introducing the SFSP concept to users to at least define solid Input Rules for them. Adding this concrete piece may initially help them get used to using the SFSP method.

Summary

As the previous conceptual walkthrough demonstrated, SFSP is a mechanism where users combine an easy to remember number with a descriptor of the password's use-case in a predetermined manner. This results in strong passwords that users can easily and accurately reproduce even if they do not remember the password itself. They can do this without having to write down passwords for future reference. As opposed to a typical password, which is simply something you know, SFSP is a combination of three things you know:

SFSP Things to Know

- Something you know – your standard secret password component (Secret Code)
- Something you know – your standard method of password creation (Input Rules)
- Something you know – your description of what you are doing/using (Memory Cue)

Illustration V

The SFSP formula is composed of three major parts, Input Rules, a Secret Code, and a Memory Cue. Once set, the Input Rules remain static. They are the instructions for how a Memory Cue is mixed with a Secret Code. They also dictate which special characters are used and where. Moreover, they provide the variable parts of the password so that mandatory password changes and forced password resets do not make passwords difficult for the user to manage.

Secret Codes are numbers to be mixed with the Memory Cue as the Input Rules dictate. They are also static. The user picks one Secret Code to be permanently used across all passwords. This code is not revealed to anyone, and users are expected to pick something that they are sure to always remember, even if it is guessable (Remember, even if someone guesses a user's Secret Code they will still not know the Input Rules

that the user implemented, nor will they know the Memory Cue that the user has associated with a particular account.

The Memory Cue is a word that the user picks that he or she associates with a password use-case. It can be the name of the application, website or document being accessed. Strong passwords can be created even when the Memory Cues picked are obvious ones, but best security practices encourage users to pick non-obvious Memory Cues *as long as* what they pick will be something that always comes to mind when the think of the password use-case that the Memory Cue is to describe.

All parts of SFSP are user defined and based upon personal experience and preference. Thus users' SFSPs and SFSP passwords are unique (for all practical purposes). This feature means that users can pick password components that are easy for them to remember while still having strong passwords. With password guessing (inference attacks) there is only one component that needs to be guessed, and that is the single word that makes up the password. Psychologists claim that creating passwords has become a type of personality test as users attempt to find passwords that tend to describe themselves relative to whatever task they are performing. According to research findings on this subject, passwords tend to be quite guessable. As more is known about a person the more guessable their password. If the person and the person's password policy are known (are numbers and special characters required), then the password gets even easier to guess (Lee) .

This does not apply to SFSP. SFSP passwords are not easily guessable. With SFSP the Secret Code, the Memory Cue *and* all of the Input Rules all need to be properly guessed if an inference attack is to be successful.

Examples of properly constructed SFSP passwords – Next Page:

Examples of properly constructed SFSP passwords – Next Page:

SFSP FOR THE WORKPLACE, WITH DATE CODES AS THE VARIABLE CHARACTERS

Secret Code
9161996 (Shirley's wedding date – September 16, 1996)

Rule Number	Rule Description
R1	• Insert Secret Code after third character of Memory Cue
R2	• Surround the Secret Code with the special character of your choice (Shirley picks # as her special character)
NEW ▶ R3	• Only capitalize the first character of the Memory Cue and the first character after the Secret Code
R4	• Truncate using only the first five letters of Memory Cue
NEW ▶ R5	• Add the current month and year to the end of everything you just typed in the following format MMYY (ex. "0105" for January 2005)

Current Date = March 2005

Memory Cues: Dependant upon use-case. See below:

Use Case (Password Application)	Resulting SFSP Password
1. Document named, BUDGET	• Bud#9161996#Ge0305
2. Spreadsheet named, Expense List	• Exp#9161996#En0305
3. Windows Login (Cue = Windows)	• Win#9161996#Do0305
4. Hotmail Account	• Hot#9161996#Ma0305
5. Yahoo Account	• Yah#9161996#Ho0305
6. Unatime Web Timesheet	• Una#9161996#Ti0305
7. Intranet Account	• Int#9161996#An0305

After 90 days, in June passwords must change. So these passwords will become:

Use Case (Password Application)	Resulting SFSP Password
1. Document named, BUDGET	• Bud#9161996#Ge0605
2. Spreadsheet named, Expense List	• Exp#9161996#En0605
3. Windows Login (Cue = Windows)	• Win#9161996#Do0605
4. Hotmail Account	• Hot#9161996#Ma0605
5. Yahoo Account	• Yah#9161996#Ho0605
6. Unatime Web Timesheet	• Una#9161996#Ti0605
7. Intranet Account	• Int#9161996#An0605

Once Shirley gets accustomed to typing her SFSP password, all we are asking of her is to always insert #9161996# (her wedding date) in the middle of her account name. Then she tacks a date on the end and that is easy to remember. Keep in mind however that "in the middle" will have different meanings depending on each person's Input Rules. It could be that the Secret Code and special characters are inserted before or after any account name character. It could even be intermingled as alternating characters with the account name.

Illustration W

iSFSP FOR INTERNET SITES, WITH SEQUENCE CHARACTERS AS THE VARIABLE CHARACTER

Secret Code
9161996 (Shirley's wedding date – September 16, 1996)

Rule Number	Rule Description
R1	• Insert Secret Code after third character of Memory Cue
R2	• Surround the Secret Code with the special character of your choice (Shirley picks # as her special character)
R3	• Only capitalize the first character of the Memory Cue and the first character after the Secret Code
R4	• Truncate using only the first five letters of Memory Cue
NEW ▶ R5	• Add a sequence number to the end of everything you just typed (ex. "1" or "A")

Current Date = March 2005 (this has no impact on sequence based SFSP passwords)

Memory Cues: Dependant upon use-case. See below:

Use Case (Password Application)	Resulting SFSP Password
1. Document named, BUDGET	• Bud#9161996#Ge1
2. Spreadsheet named, Expense List	• Exp#9161996#En1
3. Windows Login (Cue = Windows)	• Win#9161996#Do1
4. Hotmail Account	• Hot#9161996#Ma1
5. Yahoo Account	• Yah#9161996#Ho1
6. Unatime Web Timesheet	• Una#9161996#Ti1
7. Intranet Account	• Int#9161996#An1

After a forced password change/reset these passwords would become:

Use Case (Password Application)	Resulting SFSP Password
1. Document named, BUDGET	• Bud#9161996#Ge2
2. Spreadsheet named, Expense List	• Exp#9161996#En2
3. Windows Login (Cue = Windows)	• Win#9161996#Do2
4. Hotmail Account	• Hot#9161996#Ma2
5. Yahoo Account	• Yah#9161996#Ho2
6. Unatime Web Timesheet	• Una#9161996#Ti2
7. Intranet Account	• Int#9161996#An2

(Note: many internet sites have character limits on their passwords. It may be necessary to write your Secret Code in a truncated/shortened form, or to pick a short Secret Code in order to comply with these rules. In the example above Shirley could shorten her Secret Code from 9161996 (9/16/1996) to 91696 (9/16/96), 91996 (9/1996), or simply 996 (9/96).

Illustration X

Issues

I believe that SFSP is a solution that addresses all of the most prevalent password related problems. There is an issue however (see notes under Illustration C), that we have not yet fully addressed, which is not limited to the password space yet still compromises the effectiveness of password based authentication. I am speaking of *Sniffers*. Sniffers are tools that are used to covertly record information that is being processed by computers or which is in transit to or from them. Some of the most secure

authentication methods are vulnerable to sniffing. Even the robust Kerberos ticket based authentication system has given up sensitive information to sniffers.

Sniffing

There are two major types of sniffing, network centric and system centric. *Network centric* recorders are usually referred to as network sniffers. Network sniffers listen to traffic traversing a particular network link or segment, but cannot, for the most part, listen across switched networks without first compromising (legitimately or otherwise) at least one network switch (there are however techniques for getting some information across properly functioning switches. For example, the product *Angst* is a simple active sniffer that floods the local network with random MAC addresses, causing switches to send packets to all ports) (Security Focus) . The two types of network sniffers are passive and active. Passive versions of sniffers simply record all traffic that they see, while active sniffers (A.K.A. scanners) probe networks for information.

System centric sniffers are usually referred to as loggers. Loggers most commonly come in the form of keystroke logging tools. But there are also activity, transaction and service/DLL loggers, to name a few. Loggers can be hardware or software devices.

Network Centric Sniffing

Does SFSP enter into the sniffing equation? Absolutely. Take *Network centric* sniffing for example. There are two major issues in network traffic that contribute to the compromise of passwords by sniffers. They are plain text passwords and hash matching. Plain text is a phrase to describe when information, in this case a user's password, is sent across a network in a clearly readable format. Some applications natively use plain text, like FTP (File Transfer Protocol) and POP (Post Office Protocol) protocols. SFSP is not the answer to plain text transport issues. While it is more difficult to recognize a SFSP password with the naked eye, many sniffers will identify and highlight the password if it is sent in plain text. The answer to this problem is to avoid sending in plain text. This means using versions of products that encrypt their transmissions, or using a transport medium like IPSEC which encrypts all traffic before sending it. (A version of network traffic encryption, PPTP is included in Windows – Windows VPN - that can be used to encrypt plain text traffic for free if you have some applications that must send/receive in plain text).

However, even when passwords are encrypted sniffers can be a problem. If users have weak passwords (dictionary words or small character sets) then the hash marks that encryption creates from encrypting a password can be matched against hash tables. Hash tables, like the Rainbow Tables (Kahn) pre-encrypt all possible character combinations and produce a table with the resulting hash marks. This is called a pre-computation attack. Sniffers can find hash marks traversing networks and these can be compared against hash tables. When a matching hash is found on the table, the plain text password used to create the hash is also listed, revealing the users true password.

On the other hand, if users use strong passwords, then the resulting hash is too long to be calculated (and stored in a table) by normal means and thus will not be found in a

hash table. In fact strong passwords cannot be cracked with current technology within our lifetime. That is how SFSP will help defend against network sniffing. It will provide strong passwords that cannot be cracked in reasonable times, even if the hashes are stolen from the network with sniffers.

Consider this; the average length of SFSP passwords falls between 10 and 14 characters. Assuming the maximum possible number of characters is being used (as is the case with SFSP - 94 characters represents all displayable characters including mixed case letters), a password of only 3 characters could be cracked (deciphered/unencrypted) with current personal computing technology in less than 9 seconds. It is safe to assume that hash marks of all possible three character passwords are available to those wishing to crack them. A similar password of 4 characters could be cracked in less than 15 minutes (Shaffer). Passwords created from all possible character sets (upper and lower case letters with numbers and special characters) are considered *strong* by NSA if they have 8 characters (8 chars sufficient but 12 recommended) (Guide to Securing…) . The following illustration shows the average time it would take to crack strong passwords of varying lengths using only a Pentium2® 800 MHz computer:

Password Length (Characters)	Time Needed to Crack Password	Units of Time
3	8.3	Seconds
4	13	Minutes
5	20.4	Hours
6	2.63	Months
7	20.6	Years
8	1.93	Millennia
9	182	Millennia
10	17,079	Millennia
11	1,608,461	Millennia
12	150,913,342	Millennia

(Shaffer)
Illustration Y

Current PC technology is considerably faster than what is represented in the previous Illustration (Y), which assumed a processing rate of 100,000 transactions/cycles per second. Today the fastest password cracking tools can perform nearly 1,000,000 transactions/cycles per second.

However fast that seems, it would still take 900 years to crack an 8 character SFSP password and 15 trillion years to crack a 12 character SFSP password. Therefore, the combination of SFSP and network traffic encryption is a viable defense against network sniffing.

System Centric Sniffing

What about *System centric Sniffing*? Can SFSP defend against keystroke logging and system monitoring? The answer is no. No password or password methodology can. But there is a simple solution that should be used with SFSP to protect systems. Most solution components used to address keystroke logging already exist on systems with reasonable security measures in place. Many software solutions combine Firewall, IDS (Intrusion Detection), IPS (Intrusion Prevention), and Spy Ware Prevention/Removal as part of the *technical solution* to keystroke logging. There are also network measures that will help recognize and block the activities of loggers. But as with network sniffing, where the total solution is part *technology* (network encryption) and part user action (SFSP), the solution for logging is also part technology and part user action.

The user side of the key logging solution is what I call Defensive I/O (Input/Output). In this discussion I will focus on the inputting side of this Defensive I/O. While Defensive I/O is not part of SFSP, they are very complementary. I include this brief discussion of Defensive I/O because it, combined with the other simple recommendations in this paper, provide a very secure authentication process that can be implemented and preserved without adding or changing costly infrastructure.

For purposes here Defensive I/O simply refers to inputting information in unconventional ways so as to fool anyone that is able to record some of a user's data before the recording mechanism is discovered, stopped and removed. Defensive I/O, like SFSP is very simple. It means inputting information out of sequence, and using the mouse rather than the keyboard to move between data entry fields (this may not be practical in intense data entry environments, but will be helpful to any user when operating on the internet).

Keystroke Logging

Loggers come in various flavors. Once a password is entered into a properly configured system using SFSP it is very difficult for an unauthorized user to get a system to reveal that password. Because of this, a popular type of logger known as a keystroke logger, attempts to steal passwords and other information as they are entered into a system, before it is protected (encrypted/hashed). We will focus on this flavor of logger as it is the remaining weakness in the password equation being discussed.

Keystroke loggers come in the form of hardware adaptors that fit in-line with the keyboard cable and software/applications that run on the host machine. Most of them are capable of recording all keystrokes, some can also record special keystroke combinations (that are used to augment inputted passwords – for example, hold down the Ctrl or Alt key while inputting a password). Some of these can email their stolen data out to the internet while others allow intruders to come into the user's machine and view the data in place. There are also a few that I call Advanced Keystroke Loggers. I will discuss those momentarily.

For the most part keystroke loggers can be defeated by Defensive I/O. This is how keystroke loggers work. If I were using a system that was being monitored by a keystroke logger, and I typed:

"I am going shopping.
I will be back soon."

Then the keystroke logger would create a log file (a file in which to record all of my information) and it would record:

I am going shopping.[RETURN]I will be back soon.

If you opened the log file you would see the exact text that I typed and special function keys notated that I pressed (in this example the special function key was the return key, notated as "[RETURN]". See Illustration Z below for more examples:

Example A

1) Words typed:	House
2) *Logger Records*:	House
1) Words typed:	Housse → corrected with backspace to house
2) *Logger Records*:	Housse[BACKSPACE] [BACKSPACE]e

Example B

1) Words typed:	Housse → corrected with mouse highlight of "s" and delete key to house (ex. Housse→Housse – hit delete key→House)
2) *Logger Records*:	Housse[DEL]

Notice with Example A you can look at the Log File and clearly see ("House"), or clearly reproduce (Housse) the inputted text. Look at the misspelling of house in the second part of Example A. If I follow what is in the Log File I would type: "H", then "o", then "u",

then “s”, then “s”, then “e”, then the backspace key, then the backspace key again, then the “e”, and I will have the originally input text of “House” remaining.

This is how normal inputting is revealed with typical keystroke logging. However, look at Example B. If I type exactly what is in the Log File, like I did in Example A, I would type: “H”, then “o”, then “u”, then “s”, then “s”, then “e”, then the delete key (which has no effect on the word just typed). This would leave me with the word, “Housse”, which is not what I had actually entered. I corrected the word “Housse” to the word “House” by highlighting one of the “s” characters that I typed and then hit delete. My final text therefore was “House” instead of “Housse”. I have in effect fooled the Keystroke Logger as to my final/intended text. This is the foundational principal of Defensive I/O.

Had I been entering my password it would have worked in the same fashion. For examples of good and bad data entry habits see below:

Example C - Normal Data Entry

Login name: jsmith
Password: viper

When Logging in:

First type login: jsmith (then press TAB key to go to password field)
Second type password: viper

Key Logger Records: jsmith[TAB]viper

Key Logger Translation:

Login: jsmith
Password: viper

Effect: CREDENTIALS COMPROMISED!

(**THIS IS BAD!** The Key Logger results are easy to interpret and reproduce. Since the Key Logger also recorded which website you typed to get to the password prompt, others may now go to the same site and enter your credentials for access to your account).

Example D – Reverse Order Data Entry

Login name: jsmith
Password: viper

When Logging in:

First type password: viper (then press TAB key to go to password field)
Second type login: jsmith

Key Logger Records: viper[TAB]jsmith

Key Logger Translation:

Login:	viper	OR	Login:	jsmith
Password:	jsmith		Password:	viper

Effect: CREDENTIALS COMPROMISED!

(**THIS IS BAD!** Though the Key Logger results are not as easy to interpret and reproduce, because anyone reading the log would assume that login came first and then the password, if they tried to login your account with “viper” as the login and “jsmith” as the password it would fail. However, most people trying to break into your account would try switching them. It does not help that “jsmith” looks like a login name).

Example E - Mixed Data Entry

Login name: jsmith
Password: viper

When Logging in:

First type half the password: vip (then press TAB key to go to login field)
Second type half the login: jsm (then press SHIFT-TAB key to go back to password field)
Third type next half of password: er (then press TAB key to go back to password field)
Fourth type next half of login: ith

Key Logger Records: vip[TAB]jsm([Shift][TAB])er([Shift][TAB])ith

Key Logger Translation:

Login:	viper	OR	Login:	jsmith
Password:	jsmith		Password:	viper

Effect: CREDENTIALS COMPROMISED!

(**THIS IS BAD!** The Key Logger results are VERY EASY to interpret and reproduce. Anyone reading can see that you typed a little and tabbed (jumped) around. With very little effort they can reproduce your input and access your online account).

Example F - Mixed Data Entry, with mouse

Login name: jsmith
Password: viper

When Logging in:

First type half the password: vip (then use the mouse to go to login field)
Second type half the login: jsm (then use the mouse to go back to the password)
Third type next half of password: er (then use mouse to go to login)

Fourth type next half of login: ith

Key Logger Records: vipjsmerith

Key Logger Translation:

Login:	vipjsmerith OR	Login:	???
Password:	???	Password:	vipjsmerith

Effect: CREDENTIALS SECURE

THIS IS GOOD! The Key Logger results are NOT easy to interpret and reproduce. Anyone reading the log might assume that you were using the "remember my login" settings in Internet Explorer and that you only typed your password. They would be wrong. If they figured out or guessed that you used the mouse to move from the login to the password field then they would have to guess what login and password combination could come from the letters "vipjsmerith". That would not be intuitive or easy).

Example G - Data Entry, false characters

Login name: jsmith
Password: viper

When Logging in:

First type login: j**ohn**smith (then use [TAB] to go to password)

Second type password: viper**111**

Third use mouse to highlight the false login characters "ohn" and hit [BACKSPACE] once to erase them all

Fourth use mouse to highlight the false password characters "111" and hit [BACKSPACE] once to erase them all

Key Logger Records: **johnsmith[TAB]viper111[BACKSPACE] [BACKSPACE]**

Key Logger Translation:

Login: johnsmith (INCORRECT LOGIN)
Password: viper1 (INCORRECT PASSWORD)

Effect: CREDENTIALS SECURE

(**THIS IS GOOD!** The Key Logger results ARE easy to interpret and reproduce, but they are wrong. Anyone reading the log would assume that they have captured and interpreted the correct information, yet they would not be able to access your account, and may assume you have recently changed your credentials.
Note: you may be able to further strengthen your internet account security by making your login name adhere to the same input rules as your password).

Illustration Z

As you can see, mixing the data during data entry, adding false characters, and using the mouse to move between data entry fields are solid defenses against the average Key Logging device or program. It is most effective to use all three of these methods concurrently when entering login credentials for sensitive internet accounts. Make a habit of it. Some Keystroke Loggers only wake up and start working when you go to

certain sites (usually bank and other financial sites). This method protects against most of those too.

Having described the average Keystroke Logger I will also mention advanced Keystroke Logging systems. There are a few Keystroke Logging applications that I have tested that do not fall into normal categories. These offer typical logging (as I have covered here), remote install, remote access, emailing of all keystrokes, all emails, all chat sessions AND will take screen shots under each mouse-click, capture passwords and login information as registered and submitted by the browser, while triggering on keywords and websites. These are not the norm, though they may become the norm at some point. I have not yet tested these programs in an SSL or IPSEC environment to see if they can still capture passwords as they are submitted by the browser. But they are very powerful and having these installed on your or your users' machines equates to a fully compromised machine in the network.

As is the case with any backdoor Trojan that can allow intruders full control to a system, once a system is fully compromised users prudence will not salvage the system. The host machine and the network it is in need to have measures in place that protect against and deal with Trojans, Patch Management, and Configuration Management. If those things are not in place then even multi-factor authentication will fail to make system and network authentication reliable. Microsoft Corporation has put this concept in writing. They call it their 10 Immutable Laws of Security. They stress that if you allow someone else to install to or change your computer, then it is no longer your computer (Microsoft) .

Conclusion

Good passwords really can work as the basis of a good authentication system. The biggest problems with passwords are users. They will either use weak passwords or improperly store and forget strong ones. The other main problem with passwords is that they can be cracked (compromised). SFSP offers a solution for all of these; easy to remember passwords that are too strong to be compromised in a reasonable timeframe. SFSP protects against Cracking, Pre-computation Attacks, Guessable Passwords, Improper Password Storage, and Stolen Hashes while providing an easy defensive alternative against Null and Default passwords.

I have heard SFSP explained as simply as this, "You basically take your easy to remember password and stick your birth date in the middle of it surrounded by quotes. Then add the date to the front of it and you're done." This is an accurate description. It is not the only way to implement SFSP, but it does transmit the concept rather well in plain English. It also produces strong passwords from nearly any standard password, even from something as simple as the word "Pass" (01#Pa4141971ss – where the date is January and the birth date is 4/14/1971).

What is the bottom line? It is this, with simple user tools like SFSP and Defensive I/O, and common best practice IT security, password based authentication can deliver

reliable and strong results without infrastructure upgrades. SFSP is the major component of the model that I am proposing and it can be implemented with very little user training and little to no funds allocation.

> A good password is more than just a complex password. A good password is one that is not easily guessed but still easy to remember. It should be long and should consist of letters, numbers, and symbols, but still easy to type quickly with few errors. It should have elements of randomness that only a computer can provide while still having familiarity that only a human can provide.
>
> But the best password of all is the one that the user chooses based on an educated understanding of passwords - a password that is hard to crack, but never forgotten. And the best password policy is one that helps users in creating these passwords. (Security Focus, Ten Password Myths)

SFSP Problem Resolution Matrix

Common Problem	Common Solution
1. Guessable Passwords	• Strong Passwords through SFSP
2. Cracking Passwords	• Strong Passwords through SFSP
3. Pre-computational Attacks	• Strong Passwords through SFSP
4. Weak Passwords	• Strong Passwords through SFSP
5. Null Passwords	• Strong Passwords through SFSP
6. Default Passwords	• Strong Passwords through SFSP
7. Improper Password Storage - Writing Down Passwords	• Easy to remember Passwords (through SFSP)
8. Clear Text/Plain Text Passwords	• Don't send plain text. Encrypt tunnels or upgrade to applications that support encryption – Windows VPN (PPTP) is included free in Windows
9. Weak Password Hashes	• Strong Passwords through SFSP - makes hash marks too large and complex for rainbow-like tables to be able to calculate them and takes too long to crack (lifetimes)
10. Keystroke Logging	• Anti-spyware software, profile enforcement and policies on configuration changes and Defensive I/O

Illustration AA

Appendix A

How to create a SFSP logon password.

A.

1. Pick any special character. You will always use it for your passwords **(**like !@#$%^& (*+)=-;:'"~`][} {\|><?/.,`**)**
2. Pick a secret 3 or 4 digit number (could be birthday, like April 5, 1956 or 4/5/56, written without the slashes= 4556 (this is your secret code)
3. Pick a very simple password that you can remember (this is the root of your password – it can be the name of the application/site you are logging into, such as Windows, Hotmail, MS Word, Resume, etc.)

B.

1. Always surround your root password with your favorite special character
2. Always insert your special number after the second character of your root password
3. Always capitalize the first character **after** your secret code (now you have the unchanging part of your password – the **Static Password**)
4. (Now for the part of your password that changes every 90 days when you are forced create a new password) Always add the creation date to the end your new static password. Add it as a combination of the calendar quarter plus the calendar year (Quarter 1, or Q1 of 2005 would be 1 and 2005 written together as 12005 – Now you have your **Full Password**)

Example 1 (Password = ~Ti4556M~12005)

Step 1	~
Step 2	4556
Step 3	Tim
Step 4	~Tim~
Step 5	~Ti4556m~
Step 6	~Ti4556M~
Step 7	~Ti4556M~12005

Example 2

(Use the same technique for every account that you have. Just change the root password to the name of the account or application and everything else is the same)

hotmail (Password = **~ho4556Tmail~12005**)

Step 1	~
Step 2	4556
Step 3	hotmail
Step 4	~hotmail~
Step 5	~ho4556tmail~
Step 6	~ ho4556Tmail~
Step 7	~ho4556Tmail~12005

Windows (Password = **~Ti4556M~12005**)

Step 1	~
Step 2	4556
Step 3	windows
Step 4	~ windows ~
Step 5	~wi4556ndows~
Step 6	~wi4556Ndows~
Step 7	~wi4556Ndows~12005

Works Cited

Burnet, Mark. "Ten Windows Password Myths." March 7, 2002. March 2005. <http://www.securityfocus.com/infocus/1554>.

Hensing, Robert. "The Future of Passwords." August 2001. February 2005. <http://blogs.msdn.com/robert_hensing/archive/2004/08/23/218903.aspx>.

Hensing, Robert. "Why you shouldn't be using passwords of any kind on your Windows networks ." September 2004. February 2005. <http://blogs.msdn.com/robert_hensing/archive/2004/07/28/199610.aspx>.

Internet.Com Corporation. Webopedia website. March 2005. <http://www.webopedia.com/TERM/s/strong_password.html>.

Kahn, Ramius. The Rainbow Tables. February 2005: <http://www.rainbowtables.net>.

Lee, Jennifer. "And the Password Is…Waterloo." December 27, 2001. February 2005. <http://www.securityfocus.com/archive12/247456>.

McAuliffe, Wendy. "Computer Passwords Reveal Workers' Secrets." June 28, 2001. February 2005. <http://news.zdnet.com/2100-9595_22-530187.html>.

Microsoft Corporation. "10 Immutable Laws of Security." 2005. March 2005. <http://www.microsoft.com/technet/archive/community/columns/security/essays/10imlaws.mspx>.

Microsoft Corporation. "Microsoft Security Glossary". 2005. March 2005. http://www.microsoft.com/security/glossary.mspx#s (November 19, 2004)

Microsoft Corporation. Microsoft Security Quiz. 2005. March 2005. <http://office.microsoft.com/en-us/assistance/QZ011127521033.aspx>.

National Security Agency. "Guide to Securing Microsoft Windows XP" December 1, 2003. February 2005. <http://www.nsa.gov/snac/os/winxp/winxp.pdf>.

SANS InfoSec Reading Room. Mortensen, Jason. "Password Protection: Is This the Best We Can Do?" August 2001. March 2005. <http://www.sans.org/rr/whitepapers/authentication/114.php>.

SANS Top Twenty. SANS Top Twenty Webpage. "W5 Windows Authentication." March 2005. <http://www.sans.org/top20/>.

Security Focus. "Ten Windows Password Myths." March 7, 2002. February 2005. < http://www.securityfocus.com/infocus/1554>.

Security Focus. "Tools Archive". October 21, 2001. February 2005. <http://www.securityfocus.com/tools/category/4>.

Shaffer, George. Geodsoft Website. 2004. February 2005. <http://geodsoft.com/howto/password/password_basics.htm>.

University of Illinois. "Business and Financial Policies and Procedures" June 14, 2004. February 2005. <http://www.obfs.uillinois.edu/manual/central_p/sec19-5.htm>.

University of Illinois. Definition of Strong Password. February 2005. <http://www.cio.uiuc.edu/policies/passwordpolicy.html>.

University of Minnesota. "Tips for Choosing a Password." 2003. February 2005. <http://www1.umn.edu/oit/security/passwordguide.html>.

University of Minnesota. Online Help. September 3, 2004. February 2005. <http://www.cs.umn.edu/help/security/password-selection.php>.

Last Updated: October 18th, 2017

Upcoming SANS App Sec Training

SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - DEV522: Defending Web Applications Security Essentials	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, Netherlands	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Oslo 2018	Oslo, Norway	Feb 05, 2018 - Feb 10, 2018	Live Event
Community SANS Indianapolis DEV534	Indianapolis, IN	Feb 05, 2018 - Feb 06, 2018	Community SANS
Cloud Security Summit & Training 2018	San Diego, CA	Feb 19, 2018 - Feb 26, 2018	Live Event
Communtiy SANS Seattle DEV534	Seattle, WA	Feb 26, 2018 - Feb 27, 2018	Community SANS
San Francisco Spring 2018 - DEV522: Defending Web Applications Security Essentials	San Francisco, CA	Mar 12, 2018 - Apr 17, 2018	vLive
SANS San Francisco Spring 2018	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced