Rethinking Password Policies

Abe Singer, Warren Anderson, Rik Farrow

August 2013

"In the practice of security we have accumulated a number of "rules of thumb" that many people accept without careful consideration. Some of these get included in policies, and thus may get propagated to environments they were not meant to address. It is also the case that as technology changes, the underlying (and unstated) assumptions underlying these bits of conventional wisdom also change. The result is a stale policy that may no longer be effective…or possibly even dangerous."

— Gene Spafford [23]

We are all familiar with having "rules" for passwords: must have at characters from various character sets, have a minimum length, get changed regularly, not be written down, etc.

These rules are supposed to make passwords "secure," but there's little to no research to support that argument. In fact, they can even weaken security. Most of the "best practices" in use today are based largely on folklore, or in some cases on severely outdated theories of password strength. Even the US Government "standards" on password strength appear to be based on nothing more than then-current default settings on a particular operating system.

These password best practices have several usability problems. Some believe that security and usability are mutually exclusive, and therefore security has to make things difficult. We argue that security *depends* on usability.

Passwords have to be strong enough to defeat cracking attempts, yet usable. This requires both an understanding of usability, and quantitative measurements of password strength.

Below we provide a summary of the relatively scant research and government standards that have led to where we are today, an overview of usability as it applies to security, an analysis as to how current best practices aren't effective, and finish with a modest proposal for more usable and secure passwords.

Why do we have password rules?

An attacker can try to discover passwords using *guessing* attacks, whereby the attacker tries various passwords to find one that matches that of the target. The simplest form of such attack is a *brute-force* attack — trying every possible password (the entire *key space*). This approach guarantees that the attacker will eventually find the password, if the attacker has enough compute resources to do so.

Users, left to their own devices, tend to choose passwords using real words and simple variations of them. Which is understandable — users want to have a password that's easy to remember. Attackers, knowing this, use dictionaries of real words for dictionary attacks (*cracking*). If the dictionary is considerably smaller than the keyspace, the computational cost of the attack is much smaller than that of a brute force attack.

Password strength rules ostensibly force the user to choose a password that's not in the attacker's dictionary. More formally, the rules attempt to prevent successful *dictionary* attacks, by ensuring that users choose passwords with sufficient *entropy* to render the attack infeasible. Entropy is the measure of the probability distribution of the passwords across the keyspace; a measure of the relative randomness of each password to all the other passwords.

Note that password strength rules provide no protection from *brute-force* attack. (To be clear, a brute-force attack is an exhaustive attack against the entire keyspace, a dictionary attack is an attack against a smaller subset of the keyspace).

So how did password rules evolve into the current "best practices?" Here's a brief history.

Some history of password attacks, and rules.

"The problem with using passwords that are derived directly from obvious words is that when a user thinks "Hah, no one will guess this permutation," they are almost invariably wrong. Who would ever suspect that I would find their passwords when they chose "fylgjas" (guardian creatures from Norse mythology), or "pataitai" (the Chinese word for "hen-pecked husband")?

— Dan Klein [24]

Passwords for computer use date back to at least the 1960s. The first mention of computer passwords in open literature was on the MIT Compatible Time Sharing System (CTSS), which was one of the first (if not the first) multi-user operating systems. At that time, passwords were used to separate and identify users, in order to control users' use of limited resources such as CPU time.

The desire and attempts to compromise passwords has been around about as long. A graduate student on the CTSS system, who needed more compute time than allocated, admitted to have taken advantage of a bug in the system to obtain a copy of the password file, which was not encrypted/hashed. [25]

Dictionary attacks were first discussed in 1979. [8] At that time, the passwords were hashed, and the hashes were kept in the passwd file and readable by all users. Morris and Thompson acknowledge the threat of dictionary attacks, using a dictionary of English words.

[8] also proposed the first example of password rules, where users were required to choose a password of at least a minimum length (6 characters for alphabetic, 5 characters if the password included non-alpha). They alternatively proposed system-generated random passwords while noting a case where that approach actually weakened password security.

The "Green Book" [9] discusses strength of crypto algorithms, and includes suggestions for password length and generation.

On the latter point, we note that it specifically recommends that password be machine generated due to users choosing bad passwords. The Green Book also recommends that users memorize their passwords, but allows for writing down passwords as long as the written password is sufficiently secured.

While dictionary attacks were discussed as far back in 1979, the first openly published program to do so was Crack, posted to Usenet by Alec Muffet in 1992. [10] Discussions about the posting acknowledged that underground tools had been available to attackers for years. Around the same time, Bellovin notes that attackers had been observed trying to download the password file from Bell Labs ftp servers. [26]

Crack was intended for use by system administrators to run a dictionary attack against the hashed passwords on their systems, to detect and change weak passwords hopefully before an attacker could do so. Muffet and others also acknowledge that a preferred approach would be to prevent users from entering dictionary words in the first place, but tools to do so were not yet available, and vendors were slow to implement such features (Indeed, it was more than a decade before such capabilities commonly available).

On a side note, we found it interesting in retrospect that the publication of Crack was quite controversial at the time. Many people thought it wrong to release a "hacker tool" that could be used by attackers. The counter argument was that the attackers already had the tools, and the publication of Crack provided defenders with a tool to fend off attackers. These days, tools such as Crack are considered a standard component of a defender's toolbox, and there is no controversy about it.

In 1989-1992, Klein [7][24] ran dictionary attacks against a set of ~15,000 encrypted passwords contributed by volunteers. Armed with a ~62,000 word dictionary, Klein cracked approximately 39% of the passwords.

At that point in time, the Unix and other password entry systems did not impose any restrictions on what password a user could choose; obtaining strong passwords was expected to get addressed with user training. Klein and Bishop proposed "pro-active password checking," whereby the password setting program would check the password entered by the user against a dictionary and for certain compositional aspects such as upper/lower case.[7]

A similar, but less rigorous, proposal was made in [12]. The authors not only performed password cracking, but pre-computed hashes for dictionary words and stored them, so that successive cracking attempts could be done without having to recompute hashes each time. At the time, due to I/O limitations, their approach made cracking 28 times faster than computation.

In 2003, the Teracrack project repeated the pre-computation experiment with much larger dictionary, and for all 4096 salts used for Unix password hashes. The results took up 1.5 Tb of storage, and reduced the time to attempt to crack any one password hash to a maximum of a couple of minutes — the time required to load and read a tape from the library. The same technique today could be stored on a single hard disk and search time reduced to the disk seek time. Note that this approach only is feasible for Unix DES salts, the MD5 hashes used on newer versions of Unix/Linux use a 20 bit salts, making the storage requirements much too large in todays' environment. [13]

Standards for Password Rules

One would think, with the research done on the guessability of passwords, that the current "best practices" for password rules have been based on rigorous research and analysis, resulting in thoroughly documented standards. In reality, what few standards exist are based on inconsistent research at best, and in some cases appear to be pulled out of thin air.

NASA's password requirements are representative of the common "best practice" for password strength. [14] The document claim to be in compliance with the Federal Desktop Core Configuration. [15] However, we could find no documents within the FDCC provide specific password complexity requirements.

A search for Federal documents on password strength came up with the following data, which is itself incomplete and inconsistent:

[17] notes that users are bad at choosing passwords and recommends that passwords be automatically generated.

[18] states that "Passwords shorter than 10 characters are usually considered to be weak" and "Passphrases shorter than 20 characters are usually considered weak," but does not provide any justification or citations for those statements.

[19], notes the difficulty in measuring the entropy of user chosen passwords.

[20] mentions password composition as a factor in password requirements, but leaves the specific requirements up to the organization.

[21] does not give specific requirements for password strength, but suggests that users be trained on good password practices and that systems might restrict password choices based on password composition. It also gives suggestions for choosing good passwords such as using the first character of a well known phrase, etc.

The document also notes the difference between the keyspace of the password hashing scheme and what users actually pick: "When determining policies for password length and complexity, organizations should consider maximum and likely actual keyspace."

[22] states that "Totally alphabetic password composition should be discouraged."

The first case we could find of the current common practice of requiring characters from different character sets is from Microsoft. Microsoft Windows NT Service Pack 2 introduced password strength requirements with a minimum length of 6 characters, and the password must contain at least one character from the four character sets of A-Z, a-z, 0-9, and "special" or punctuation characters. Microsoft gives no justification or citations for those requirements, and

the requirements were hard coded.

Windows 2000 allowed for changing of the requirements, but left the default the same as in NT but with a minimum password length of 8 characters.

Password Aging

*There's no **there** there.*

— Gertrude Stein

Aging passwords — requiring users to change passwords at regular intervals — originated due to the use of hashing algorithms which were weak enough to be subject to a brute force attack. Password aging is a defense against brute force attacks, not dictionary attacks.

The Green Book [9] details the relationship between password length and password lifetime, and includes formulae for calculating minimum password length. Note that at the time that the Green Book was written, brute-force attacks against the hash algorithms in use were considered within reach of government funded agencies.

NIST SP800-118 [21] comments on password lifetime limits: "Password expiration is also often a source of frustration to users, who are often required to create and remember new passwords every month or two for dozens of user accounts. [...]

In cases where password hashes are at significant risk of compromise, organizations should take estimates of cracking abilities into consideration when setting policies for password expiration, length, and complexity."

For Windows 2000, Microsoft stated "Where security is a concern, good values [for password lifetimes] are 30, 60, or 90 days. Where security is less important, good values are 120, 150, or 180 days." [27] But they do not provide any definition for what "important" and "less important" are, nor how they calculated those numbers. Defaults password lifetimes in Windows 2000 were 42 days.

None of these recommendations provide *any* analysis as to how much, if any, password aging reduces the risk of dictionary attacks. For any given password aging interval n, assuming some unknown attack on the passwords has equal probability of discovery at any point over n, the mean exposure time for a compromised password is n/2. It would seem that for any reasonable value of n, the exposure time would be unacceptable.

We argue below that eliminating password aging contributes greatly to the usability of passwords.

Passwords and Usability

"This belief of the fundamental conflict between strong computer security mechanisms and usable computer systems pervades much of modern computing. According to this belief, in order to be secure, a computer system must employ security mechanisms that are sophisticated and complex — and therefore difficult to use."

— Matt Bishop [4]

Computing professionals have long held onto the belief of an inherent tension between security and usability, that each works against the other, which has often led to a disregard of usability for the sake of securing systems. But that belief turns out to be a misconception based largely on a lack of understanding of the meaning of usability.

So what do we mean by "usability" in the context of security? Usability is often associated with perceived ease of use — the less effort required, the more usable the system. More fundamental properties of usability are [2][3]

- Is the user able to understand what is required of her; Can the user understand how to use the security mechanism properly, recognize when she's failed, and understand why?
- Is the user capable of using the mechanism properly?
- Does the user understand the goal of the security mechanism;
- Is the user motivated to follow the security requirements?
- Do the requirements and interface match the user's understanding of the security goals?

The study of Human Factors separates tasks into "production tasks" and "supporting tasks" (sometimes called "enabling tasks.") [4]. Production tasks are the actual end goal of the user, the desired output. Supporting tasks are those that enable the user to work on the production tasks. For example, a user authenticating herself to the system enables the user to access data on the system. Accessing the data is the production task, and authentication is the supporting task. Users don't want to spend time on supporting tasks — those that have too much of an impact on production tasks affect the usability of the system, and productivity of the users.

"Ease of use" is an important factor, but does not completely equate with work factor; The work factor of supporting tasks can involve not only physical time and effort, but *cognitive load*, the measure of the ability of people to learn [5]. The amount of mental effort a user has to expend on understanding security requirements and complying with them are all a cognitive load that affects the size of the supporting task.

In order for a security mechanism to be used properly, the user must be able to understand both how and why to use it, and be able to use it efficiently. Security *depends* on usability.

The usability of common password requirements

"Does added security make things more difficult to use? Will people always resent the extra steps? The answer to both questions is the same: Not necessarily. Consider the physical world of doors and locks. Locks on houses, cars, and private records get in the way of easy access, but are tolerated because they seem necessary and the amount of effort required to open them usually seems reasonable. Note the two different components: the understanding of the necessity for protection and the reasonableness of the effort required. Both are design issues. And both require at their base, a coherent, understandable conceptual model of both the need for security or privacy and the workings of the mechanisms that enforce them."

— Don Norman [28]

Common password requirements have a negative impact on the usability of passwords. Namely:

- rules for password complexity
- requirements to change passwords on a periodic basis (password aging)
- requirements to not reuse old passwords
- prohibitions against writing down passwords

As we noted above, some of these rules were originally devised in a context that does not apply in all contexts. Here we discuss how these rules impact usability, and the ensuing risks.

Password Complexity and Aging

Password complexity rules make the user expend time and effort to devise an acceptable password, and then memorize it immediately. This imposes a cognitive load on the user and increases supporting task work factor. [2] notes the inherent cognitive conflict between the requirement that a password be effectively difficult to remember and the requirement that passwords never get written down.

Password aging further increases the cognitive load and work factor, by forcing the user to repeat the process of devising and remembering passwords repeatedly. The negative impact of this combination of rules has been noted in several places.

A study on password usage within the FAA [6] quantified the direct cost in staff time in changing passwords, noting that the costs were greatly magnified by the fact that users had numerous (up to 20!) passwords for different systems, all with different password rules and aging policies. Users were essentially in a steady-state of changing passwords.

This same study noted that due to the burden of remembering passwords, coupled with the impact of forgetting passwords on production tasks, that users adopted numerous *coping strategies*, which were in turn violation of other security policies: leaving sessions logged in, sharing passwords with coworkers, writing passwords down, etc.

Even the federal government acknowledges that password changing can cause problems. "The FIPS guidelines actually acknowledge that the load on users created by frequent password changes creates its own risks, which in many contexts outweigh those created by changing a password less frequently." [4]

And here's the fun part: there is absolutely no risk justification for *any* of the time intervals (42 days, 3 months, 6 months, 1 year) seen in current "best practices." As far as we can tell, all of those numbers have been pulled out of thin air (or less well-lit regions).

Usability of Pro-Active Password Checking

When [7] originally proposed pro-active password checking, it seemed like an effective approach: Avoiding dictionary attacks was best solved by preventing users from entering passwords that were in the dictionary. That approach assumes, of course, that one can check against a dictionary that's at least as good as any attacker would use.

Computation power in 1992 was such that a reasonably modest dictionary of 100,000 words or so, plus common substitutions, was sufficient to deter attacks. But current computational power, combined with easy on-line access to wordlists in the millions, has changed the landscape.

We made an attempt at implementing pro-active checking by doing what an attacker would do: creating the biggest dictionary we possibly could. Using 1-grams from the Google Book project. [29] We started with a list of ~4,000,000 words, and after applying the Crack substitution algorithms, ended up with a dictionary of about 90,000,000 passwords.

Having users change their passwords while checking against the dictionary was a colossal failure. There were so many unacceptable words that users became frustrated trying to come up with an acceptable password, an ended up choosing randomly until one was accepted by the system.

Pro-active password checking fails usability because it's impossible for the user to understand how to comply with the rules without guessing, and ends up increasing both the work factor and cognitive load of choosing a new password.

Risks of Writing Down Passwords

The prohibition against writing down passwords is an assumed mandatory requirement. [2] So the user is forced to devise a difficult to remember password, and then immediately remember it, further exacerbating the cognitive load on the user. [4] Add to this the often times useless feedback provided to the user while attempting to create an acceptable password.

But that risk from writing down passwords is very context dependent. Prohibition against writing passwords hails from the military, where the threat of an malicious insider (a spy) looking for written down passwords was substantial, and the liability of that risk, astronomical. That threat may be substantially lower in other contexts, where the threat of password guessing from a remote anonymous attacker is much higher.

And, as mentioned above, the burden of having to remember passwords causes users to take other measures that can impose equal or greater risks. Writing down passwords reduces the cognitive load for users, especially for passwords that get used infrequently.

Writing down passwords is also perceived as being very insecure because the passwords may get left someplace they are easy to discover. That risk can be easily mitigated with some simple rules for keeping the written password in a reasonably secure location (e.g. wallet, locked desk, etc.). Note that eve the Green Book also recommends that users memorize their passwords, but allows for writing down passwords as long as the written password is sufficiently secured.

In many environments, the risk of dictionary attacks against passwords greatly outweighs the risk of writing down passwords; strong passwords are more important than memorable passwords.

Single Sign On

The FAA study noted that many subjects had numerous passwords to remember. [6] Reducing the number of password that users have to remember greatly reduces cognitive load. A single-sign-on system, where the user has to remember and use one at a given interval (e.g. every 24 hours), has a profound effect on usability. [2]

Passwords and Entropy

People often speak of password entropy as a measurement of password strength, and attempt to measure the entropy of a given password. But as stated above, entropy is the measurement of the relative randomness of all the passwords together — you can't measure the entropy of a single password.

The only way to guarantee high entropy of user chosen passwords is to require users to enter passwords that are significantly different than other passwords. But the only way to achieve that is to reject the user's new password as being too similar to another password, which in turn provides hints about the composition of another password on the system. And the user would have no way of knowing what passwords would be considered acceptable without repeatedly trying passwords until one was accepted — a usability failure.

Password character class rules fail to provide any guarantees of entropy because they do nothing to prevent users from choosing the same or similar passwords.

Improving the Usability and Security of Passwords at the Same Time

Here's a modest proposal to make password management more usable for users, and improve the entropy of the passwords at the same time.

First, provide single/common sign on to minimize the number of passwords the user must remember (reducing cognitive load) and the number of times the user has to authenticate (minimize supporting tasks).

Allow the user to write down her password, as long as it's done in a reasonably secure manner, reducing cognitive load, and reducing the need for users to adopt insecure coping strategies.

Eliminate password aging, minimizing work factor and cognitive load for devising and remembering new passwords. Only require password change when the password may have been compromised. To minimize compromise, prohibit (or at least discourage) the users from using the same password at sites out of your control.

Eliminating aging means that you need sufficient password entropy to prevent a dictionary attack. Even if you don't eliminate aging, you need to still need to be able to quantify the entropy, in order to determine an aging interval that has acceptable risk.

So you need to implement password rules that guarantee sufficient entropy across the set of user passwords. But here's the rub: when you let users choose their own passwords, you can't devise password rules that are both usable and have enough entropy. We will be authoring a paper in the near future that demonstrates this.

One answer to that dilemma is to not let users choose their passwords, but to generate passwords for them using a random algorithm. It's an easy (perhaps only) way to assure entropy, and when done right, can be usable. At least, if the random passwords are sufficiently memorable (and typable), they can be more usable than requiring the user to choose a complicated, difficult to remember password that she can't write down and have to change often.

While the cognitive load of learning the new password *may* be greater (and we're not sure that it's true), it doesn't have to be much greater, and can be offset by allowing the user to write it down.

The above combined approach creates a "grand bargain" with the user: in return for not being able to choose her own password, the user will only have to learn this one assigned can write it down to aid with memorization, and will never (normally) have to change it.

Reasonably Memorable Random Passwords

There is a standing assertion that random passwords are difficult to remember and therefore fundamentally unusable. [4][3] However, these assertions turn on assumptions as to how those passwords get formed, e.g. random strings of characters. We argue that if done properly, they can be reasonably usable and memorable.

In order to randomly generate usable passwords, consider that not all users are the same; their criteria for acceptable passwords can vary:

- A short, complicated password requires less typing.
- A longer alpha-only password that's easy to enter via iPhone or tablet
- A very long password that's easy to remember, e.g. a pass-phrase.

Random generation of passwords can be acceptable when the user is given a set of choices within the constraints of the password entropy requirements. Giving the user a limited set of choices also gives the user the opportunity to select a password they find more memorable, reducing cognitive load.

To demonstrate, consider the following set of choices, which were generated randomly:

Passwords generated from a word list	opinion parting theological infrastructure lecture vividly
Lower case alphabetic passwords	vukizocylqhxzxiexq qgmblqmtngtiurtybj
Alpha-numeric passwords	khjd2gjact31koo7 ntrv5xbrvdbt6d05
Mixed case alphabetic passwords	ywcgyRwIdUbBsL zmbLwdAFvQuIPQ
Random passwords	im&c<Z+I)<t^ XvG[9Hm8klpN

Our experience (albeit anecdotal) with this system found that the passphrases are reasonably easy to remember.

Generating memorable random passphrases requires drawing from a dictionary of words that are already well familiar to the user. The average English-speaking adult vocabulary is 20,000-50,000 words [16], but that list includes words the user recognizes, but does not know well enough to spell or remember. Using a dictionary of the 10,000 (or less) most frequent words seems to provide passphrases that are sufficiently memorable to the user.

References

[1] National Security Agency, "Guide to the Secure Configuration of Red Hat Enterprise Linux 5." Revision 4.2. August 26, 2011.

[2] Anne Adams and Martina Angela Sasse. 1999. “Users are not the enemy.” Commun. ACM 42, 12 (December 1999), 40-46. DOI=10.1145/322796.322806

[3] Saltzer, J.H.; Schroeder, M.D., "The protection of information in computer systems," Proceedings of the IEEE , vol.63, no.9, pp.1278,1308, Sept. 1975

doi: 10.1109/PROC.1975.9939

[4] Lorrie Cranor and Simson Garfinkel. 2005. Security and Usability. O'Reilly Media, Inc.

[5] Horcher, A.-M.; Tejay, G.P., "Building a better password: The role of cognitive load in information security training," Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on , vol., no., pp.113,118, 8-11 June 2009

[6] Allendoerfer, K., & Pai, S. (2006). “Human factors considerations for passwords and other user identification techniques part 2: Field study, results and analysis.” (DOT/FAA/TC-06/09).

[7] M. Bishop and D. Klein, “Improving System Security Through Proactive Password Checking,” Computers and Security 14(3) pp. 233–249 (May/June 1995).

[8] Robert Morris and Ken Thompson. Password security: a case history. Commun. ACM 22, 11 (November 1979), 594-597. DOI=10.1145/359168.359172 http://doi.acm.org/10.1145/359168.359172]

[9] DoD Password Management Guideline, CSC-STD-002-85, Library No. S-226,994

[10] Muffet, A. (1992-4-30). “Is an apology necessary for posting Crack ? (Longish).” comp.sources.d. http://groups.google.com/group/comp.sources.d/browse_thread/thread/5a54f763d11833fc/997391d5c07758f3?#997391d5c07758f3

[12] David C. Feldmeier and Philip R. Karn. 1989. UNIX Password Security - Ten Years Later. In Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology(CRYPTO '89), Gilles Brassard (Ed.). Springer-Verlag, London, UK, UK, 44-63.

[13] Perrine, T. “The End of crypt() Passwords . . . Please?”, ;login: issue: December 2003, Volume 28, Number 6

[14] HQ Domain Passwords Frequently Asked Questions (FAQ’s), August 20, 2008. Downloaded from http://itcd.hq.nasa.gov/documents/faq_hq_domain.doc

[15] NIST Federal Desktop Core Configuraiton.

[16 Vocabulary. 24 July 2013 03:07 UTC. In *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation Inc. Encyclopedia on-line. Available from http://en.wikipedia.org/wiki/Vocabulary.

[17] Helen M. Wood. “The Use of Passwords for Controlled Access to Computer Resources.” Volume 500, Issue 9 of Computer science and technology, NBS special publication. U.S. Department of Commerce, National Bureau of Standards, 1977.

[18] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen.

“Recommendation for Password-Based Key Derivation Part 1: Storage Applications.” NIST Special Publication 800-132. National Institute of Standards and Technology. December 2010.

[19] William E. Burr, Donna F. Dodson and W. Timothy Polk. “Electronic Authentication Guideline.” NIST Special Publication 800-63. National Institute of Standards and Technology. April 2006

[20] “Recommended Security Controls for Federal Information Systems and Organizations” NIST Special Publication 800-53. National Institute of Standards and Technology. August 2009.

[21] Karen Scarfone and Murugiah Souppaya. “Guide to Enterprise Password Management (Draft).” NIST Special Publication 800-118. National Institute of Standards and Technology. April 2009.

[22] “Announcing the Standard for PASSWORD USAGE.” Federal Information Processing Standards Publication 112. National Institute of Standards and Technology 1985.

[23] Gene Spafford, “Security Myths and Passwords”, CERIAS Blog, April 19, 2006. http://www.cerias.purdue.edu/site/blog/post/password-change-myths

[24] Klein, D. V.; "Foiling the Cracker; A Survey of, and Improvements to Unix Password Security." In Proceedings of the USENIX Security Workshop, Summer 1990.

[25] Robert McMillan, “The World’s First Computer Password? It Was Useless Too.”

Wired.com. 1-27-2012. http://www.wired.com/wiredenterprise/2012/01/computer-password/

[26] Steven M. Bellovin. “There be dragons.” In Proceedings of the Third Usenix Unix Security Symposium, pages 1-16, September 1992.

[27] William R. Stanek. “Microsoft Windows 2000 Administrator's Pocket Consultant.” Microsoft Corporation. 1999.

[28] Norman, D. A. “When security gets in the way.” Interactions, volume 16, issue 6: (2010).

[29] “Google Books Ngram Viewer.” 2012. http://storage.googleapis.com/books/ngrams/books/datasetsv2.html