Do Strong Web Passwords Accomplish Anything?

Dinei Florêncio, Cormac Herley
Microsoft Research
One Microsoft Way
Redmond, WA, USA
dinei@microsoft.com, c.herley@ieee.org

Baris Coskun
ECE Department
Polytechnic University
Brooklyn, NY, USA
baris@isis.poly.edu

ABSTRACT

We find that traditional password advice given to users is somewhat dated. Strong passwords do nothing to protect online users from password stealing attacks such as phishing and keylogging, and yet they place considerable burden on users. Passwords that are too weak of course invite brute-force attacks. However, we find that relatively weak passwords, about 20 bits or so, are sufficient to make brute-force attacks on a single account unrealistic so long as a "three strikes" type rule is in place. Above that minimum it appears that increasing password strength does little to address any real threat. If a larger credential space is needed it appears better to increase the strength of the userID's rather than the passwords. For large institutions this is just as effective in deterring bulk guessing attacks and is a great deal better for users. For small institutions there appears little reason to require strong passwords for online accounts.

1. INTRODUCTION

Passwords have become the dominant means of access control to online services. With this success has come an enormous variety of attacks: each login page represents an opportunity for an attacker who is just a short sequence of characters away from someone else's email, banking, medical or social networking accounts.

1.1 Why Choose Strong Passwords?

Users are frequently reminded of the risks: the popular press often reports on the dangers of financial fraud and identity theft, and most financial institutions have security sections on their web-sites which offer advice on detecting fraud and good password practices. As to password practices traditionally users have have been advised to (*e.g.* see [3]):

- Choose strong passwords
- Change their passwords frequently
- Never write their passwords down.

Unfortunately, these recommendations appear somewhat out of date. If we enumerate the principal threats to a user's credentials they would appear to be:

1. Phishing
2. Keylogging
3. A brute-force attack on the user's account (*i.e.* an attacker knows the userID and tries to guess the password)
4. A bulk guessing attack on all accounts at the institution
5. Special knowledge or access attacks:
   (a) guessing based on information about the user
   (b) shoulder surfing
   (c) console access to a machine where password auto-fill is enabled or a password manager is in use.

As can be seen none of the password "best practices" offers any real protection against phishing or keylogging, which appear to be the most prevalent attacks. Strong passwords are just as susceptible to being stolen by a phisher or keylogger as weak ones, and changing the password frequently helps only if the attacker is extremely slow to exploit the harvested credentials.

Nonetheless, it is common to assume that stronger passwords help against guessing and brute-force attacks, *i.e.* Threats 3 and 4. We show in Section 2.1 that even a relatively weak password (*e.g.* 6 digit PIN) withstands a brute-force attack on the user's account (Threat 3), so long as a "three strikes" type lockout rule is in place. Guessing based on information about the user (Threat 5 (a)) is difficult. Further, making a password strong in the cryptographic sense is very far from making it hard to guess for someone who has knowledge of the individual. For example "Sn00py2" is a stronger password cryptographically than "749278" but might be much easier to guess for someone who knows the individual. Shoulder surfing (Threat 5(b)) does not appear to be common: humans are very good at detecting people in

their personal space and this is an attack that even unsophisticated users understand. It would be difficult to argue that users should choose stronger passwords to make the memorization task of shoulder surfers more difficult. Finally, the console access problem (Threat 5(c)) is unaffected by password strength.

Thus it would appear that the main reason for institutons such as banks to insist on strong passwords is Threat 4: *i.e.* a bulk guessing attack not just on a single account, but on many of the accounts at the same institution. We discuss this attack in detail in Section 2.2. While stronger passwords certainly make this attack more difficult they are only one tool a bank has against bulk guessing attacks. We demonstrate in this paper that a bank can lower the global break-in rate from a bulk guessing attacker while increasing usability by insisting on stronger userID's rather than stronger passwords. This is just as effective (against Threat 4) as insisting on stronger passwords but reduces the burden on users. It has no influence on susceptibility to password stealing attacks phishing and keylogging (Threats 1 and 2) or console access (Threat 5 (c)) and appears to have little effect on knowledgeable guessing or shoulder surfing (Threats 5 (a) and (b)).

2. ATTACK SCENARIOS

Passwords have long been the subject of attack. Users' habits of choosing weak and easily guessed passwords was already noted in early Unix systems [15]. While much has changed in thirty years a great deal has stayed the same: a more recent study of web password habits reveals that users often still choose the weakest passwords allowed [9]. One can view this as evidence that users are hopelessly lazy where security is concerned: in spite of frequent warnings about account fraud users on average select the weakest password they can get away with. On the other hand one can argue that users show considerable wisdom from a cost benefit standpoint: choosing a strong password generates very little benefit to a user, but it does carry considerable cost. There is little benefit in a strong password since phishing and keylogging are the main threats to a user's password and even a weak password will withstand ten years of sustained brute-force attack on an account (see Section 2.1). There is cost because strong passwords are harder to remember than weak ones, and users have many passwords to remember [9]. Since the cost (*i.e.* the difficulty of remembering stronger passwords) is borne by the user, but the benefit (increased protection against the attack of Section 2.2) is enjoyed by the bank user resistance to stronger passwords is predictable. We argue that there are better means of addressing brute-force bulk guessing attacks.

2.1 Brute-force attack on individual account

For web login servers an attacker generally does not have an offline attack on a particular account. That is, if the attacker wishes to gain access to the account userID at the login server BigBank he must attempt login. Brute-force attacks are easily detected. For example many web sites institute a "three strikes" rule whereby three unsuccessful login attempts will cause access to the account to be locked (at least for some period of time). More sophisticated rules can be applied to detect less obvious attacks; *e.g.* if the ratio of unsuccessful to successful login attempts exceeds a threshold and so on.

Thus a direct brute-force attack on the password of a given account is difficult. To consider a concrete example, if a bank allows only 6 digits PINs (a relatively weak password) and locks an account for 24 hours after three attempts an attacker could search $3 \times 365 \times 10/1e6 \approx 0.011$ or 1% of the key-space in 10 years. This seems like a small risk. Further, the ratio of unsuccessful to successful logins would be huge and hence easily detected; in reality this is a very loose upper bound on the risk of a brute-force break-in on a single account protected with a 6 digit PIN (or equivalent). In essence a brute-force attack requires searching a large portion of the password key-space. Even for a weak password, and a very active user it is easy to detect the large number of unsuccessful attempts.

2.2 Bulk guessing attack on all accounts

Thus, if even a 6 digit PIN gives good protection why do banks suggest (or demand) that users choose strong passwords? For example PayPal requires that new user passwords be at least 8 characters "is not a word you can find in the dictionary, includes both capital and lower case letters, and contains at least one special character (1-9, , , _, etc.)." Why require that the password be chosen from such a large key-space when:

- The strength of the password offers no defence against phishing, keylogging or other password stealing mechanisms
- Even a 6 digit PIN yields at most a 1% probability of success to 10 years of brute-force attack?

One reason is that PayPal, and other large institutions, must worry about attacks on more than a single user account. Suppose BigBank has 10 million online user accounts. If BigBank allows 6 digit PINs each PIN will on average be used by ten different users. Instead of searching all possible passwords for a given userID an attacker can search all possible userID's for a given password. Ten million attempts will yield ten successful logins using this strategy.

Worse, BigBank's tools to detect this type of attack are far poorer than a brute-force attack on an individual userID. When attacking the password space of a single userID it is very difficult for the attacker to conceal the attempts among the user's actual logins. Even

dispersing them in time, as we have seen, causes the unsuccessful to successful login ratio to rise. In dispersing the attack across the whole userID space however the picture changes: the ten million trials at BigBank will amount to only one unsuccessful login per account.

Only a naive attacker might try all userID with a single fixed password: this might be detected. All the attacker requires is to make ten million trials with randomly chosen PINs to harvest, in expectation, ten successful break-ins. A sensible attacker will also ensure that login attempts are launched from diverse IP addresses (as suggested in [11]). Observe that 10 million trials yield the same number of expected break-ins whether mounted against ten individual accounts, as in Section 2.1 or in a bulk guessing attack against all ten million accounts. However BigBank has excellent tools to protect against attacks on individual accounts, while a bulk guessing attack can be dispersed and appear undetectable.

2.3 Summary

We saw in the introduction that password strength had influence only over Threats 3 and 4. and 5(a) The two attack scenarios are very different however. In Threats 3 and 5(a) the attacker concentrates on one account where he knows the userID. In Threat 4 the attacker evades detection by diffusing his attempts among many accounts. But to do so efficiently he must know a large number of userID's; if he does not he must guess them. While userID is not generally considered secret the fact that obtaining a list of valid userID's is hard presents a real barrier to the Threat 4 attacker. In fact he must now search the userID-password space rather than the passwords space alone. We use this fact next.

3. CREDENTIALS

3.1 Combined userID-password search space

Suppose BigBank has $10 \cdot 2^{20} \approx 10$ million online users, and each uses a 20-bit password for access control (a 6 digit PIN is approximately 20 bits if all digits are equally likely). BigBank has simple userID's: customers are assigned consecutive 7 digit numbers (their first customer has account # 0000000 and their last # 9999999). This is approximately 23 bits. To gain entry to a BigBank account the attacker must enter 43 bits: a 7 digit (23 bit) userID and a 20 bit password. We'll call the 43-bit userID-password pair a BigBank *credential*. So the credential search space that the bulk guessing attacker lives in is 2^{43}. There are only $\approx 2^{23}$ valid accounts, so the attacker can expect to break in to one account per 2^{20} attempts.

More generally, suppose a bank, with N customers assigns b_u-bit userID's and forces users to choose a password from a password space of size b_p bits. The credential search space for the bulk guessing attacker is of size

$$2^{b_u+b_p}.$$

The attacker may want to search this space in a particular order: this is especially the case if he believes that some passwords are more likely than others. In general the bulk guessing attacker requires

$$(2^{b_u+b_p})/N \tag{1}$$

trials per successful break-in. How is BigBank to defend against this? The simple "three strikes" no longer works. In all likelihood BigBank responds as PayPal has done: force users to choose stronger passwords in an attempt to enlarge the key-space. By increasing b_p the number of trials per successful break-in, given in (1) is increased.

There are other possibilities however. In the case of BigBank we had $b_u = \log_2 N$, since the entire userID space was fully occupied (*i.e.* every single userID attempted was valid). But this assumes that the bulk guessing attacker possesses a list of every single valid userID at BigBank. If he does not he must waste trials on userID values that do not correspond to a valid account. Suppose BigBank increases the number of bits in the userID rather than the password? The effect on the size of the credential search space is the same: the attacker has a $b_u + b_p$ credential to guess. But if $b_u \gg \log_2 N$ the bulk guessing attacker will see his break-in yield given by (1) drop just as dramatically as if b_p had been increased.

3.2 Effect of institution scale: the failure of "three strikes"

The number of trials per successful break-in that a bulk guessing attacker can expect is given in (1). A key requirement is that the institution to be attacked have a sufficient number of accounts N to make it worthwhile. For example a webserver for which there are only $N = 5$ valid userID's can probably use 6 digit PINs for access without compromising security. A "three strikes" rule ensures that ten years of constant attack yields at most a 5% chance of breaking any of the accounts *even assuming that the attacker knows the userID's.* If the attacker does not have the userID list the bulk guessing attack on such a small institution becomes pointless. Thus smaller institutions can allow users to authenticate with shorter credentials than large ones.

3.3 Determining the userID list

We saw in Section 2.2 that an attacker suffered no loss of yield by attacking the combined userID-password keyspace provided he knew, or could guess valid userID's. In some cases the userID space is easy to determine, but may not be fully occupied. For example, many banks use Social Security Numbers (SSN) as the userID. In the example of BigBank only 1% of the 10^9 possible

SSN's would be valid. This means a bulk guessing attacker would waste 99 trials for every one that hit upon the userID of a valid BigBank account, assuming he has no way of determining which SSN's are actively used at BigBank. This drops his yield to one break-in per 100 million trials.

Web-sites traditionally do not confirm the validity of userID's. For example, when a login attempt fails the server commonly responds by saying that either the userID or the password is incorrect. Thus they will not confirm the validity of a given userID unless the entire credential userID-password is entered correctly. Thus, while individual users may not go to any lengths to keep their userID secret it is extremely difficult for an attacker to compile the whole list. For large institutions this offers less defence than small ones since occupancy of the userID space is high. For example, to attack a large email provider like `hotmail, yahoo` or `gmail` it is reasonable to guess that `johna, johnb`, $\cdots$, `johnz` are all almost certainly valid accounts.

Recent work by Bortz *et al.* [5] shows that confirming whether a userID is valid at a particular web-site can sometimes be revealed by timing the server's response. Generally the server takes longer to respond when a userID is valid, even if the password entered was incorrect. The method of Bortz *et al.* merely confirms that a userID is valid. An additional advantage to choosing b_u to be large is that it renders it infeasible for an attacker to build the userID list by searching the entire userID space and timing the server's response. For example, if Bigbank chooses $b_u = 40$ bits (the equivalent of a 12 digit userID) the attacker must run the timing test of [5] 10^{12} times to obtain the userID list. Even at one timing test per second this would take 34000 years.

4. DEFENCE: DIVISION OF BITS BETWEEN USERID AND PASSWORD

We have seen that to reduce the attackers yield in a bulk guessing attack the bank must increase the size, $2^{b_u+b_p}$, of the credential search space. This can be certainly be done by forcing users to choose longer passwords, but it can also be done by assigning longer userID's, provided that the attacker cannot access an accurate userID list. In either case the user must now enter more data since each users enters both a userID and password each time she logs in.

If the bank determines that the overall credential must be of a certain size (*i.e.* $b_u + b_p \geq b_{min}$) how should it divide the bits between the userID and the password (*i.e.* between the non-secret and secret portions of the credential)? We have already seen in Section 3.3 a good reason to increase b_u : it makes it infeasible to determine the entire list of userID's by timing attacks. We now argue that it also makes more sense to add bits to the userID rather than the password from a usability standpoint.

Clearly b_p must meet some minimum. But this minimum is low: it must merely protect the user from a direct brute-force on the account (Threat 3), a knowledgeable guesser and a shoulder surfing observer (Threats 5 (a) and (b)). We've seen that a 20-bit password (*e.g.* 6 digit PIN) suffices for the first. Cryptographically stronger passwords are not necessarily less guessable. Shoulder surfing does not appear to a common phenomenon.

Above this minimum value of b_p we argue that it is beneficial to increase the credential space by increasing b_u rather than b_p. By increasing b_u we make the userID's harder to remember, by increasing b_p we make the passwords harder to remember. But the userID can be written down. A user can place all of her userID's for all of her accounts in plain view without fear. That is, she can write them on a "sticky" attached to her monitor, she can store them in a readable text file, email them to herself, store them on her mp3 player or cell phone or place them in any other convenient location. She can allow the userID field to auto-complete without fear. Access to her userID list gains an attacker little (he must mount the attack of Section 2.1). Only if the attacker can aggregate the userID's of all the accounts at an institution can he benefit from the lower detection advantages of the bulk guessing attack. It is difficult to contrive a scenario where this might happen: an attacker who is in a position to spy on many users and seek out files with userID strings would get better returns by simply installing keylogging spyware.

Thus the user can store her userID in the clear so long as the bulk guessing attacker cannot aggregate across all users. The same is not true for the passwords for all of her accounts. By increasing b_p we make the passwords harder to remember. But here the user has fewer options: storing them in-the-clear carries risk from an attacker who finds the list. Thus, for a large institution if a large credential space is required, it places a smaller burden on users to assign strong userID's and allow weak passwords than weak userID's and strong passwords.

4.1 userID's: Secret or Not?

userID's have not traditionally been seen as playing any rôle in protecting against attack, and have not traditionally been regarded as secret. It is thus fair to ask if the list of all valid userID's at an institution can indeed be kept secret. Clearly web sites guard the files that contain password hashes very securely; but is the same true of the userID list? We can infer that they do, since otherwise they are open to a Denial of Service (DoS) attack that shuts down their web availability. For example, `www.fidelity.com` employs SSN's as userID's (although it allows users to choose a custom alphanumeric userID between 6 and 15 characters

(symbols and punctuation not accepted)). Suppose that an attacker gains access to the entire valid userID list (approximately 30 million customers) and that Fidelity locks an account for 24 hours after 3 unsuccessful logins. Armed with this list the attacker can lock every account with a mere 90 million login attempts. Thus, even if account break-ins from bulk guessing were not a concern, web-sites have a real need to prevent the valid userID list from leaking.

For brute-force attacks we saw that there are two attack scenarios. We assumed that the userID was known to the attacker of Section 2.1, but that the attacker of Section 2.2 could not gain access to the entire list. The need to protect against DoS attacks ensures that the web-site has a strong motive to keep the userID list from becoming public.

4.2 Lockout Strategies

In the previous analysis, we assumed the lock-out time was constant (e.g., three trials, and you're out for 24 hours). This is a common strategy, but not necessarily the best. The 1985 DoD Password management guidelines [3] suggest simply timing the password try rate, and limiting the rate to somewhere between one per second and one per minute. Other strategies might adopt a variable rate. For example, one could use a geometrically increasing lock-out time. First unsuccessful attempt and you're out for 1 second, second attempt, and you're out for 2 seconds. Third, and you're out for 4 seconds. In this way, while the consequence for a few unsuccessful trials is low, the increasing cost makes it that any brute force attack is almost infeasible. For example, after only 25 unsuccessful logins, the attacker would have to wait for 2^{25} seconds (*i.e.*, over a year) before another attempt. Besides limiting the login attempts for a particular user, the same approach can be used to limit the number of attempts from a particular originating IP address. If a compromised machine, or bot, can only be used for a total 25 attempts in a year, chances of a brute force attack are significantly reduced.

A natural question is, of course, that of DoS attack. Again here, it seems current practices are somewhat outdated. In the most common before-the-web attack scenario, the attacker would likely be trying to login from the same location as the user. In web attacks, the attacker is usually at a different location, or use a botnet of compromised machines. So, if (in the exponential lock-out time) after 20 unsuccessful trials, the attacker is locked for the next 12 days, is the legitimate account owner now also locked out for 12 days? Not necessarily. Instead of outright locking the account, a better strategy would be to have separate counters for IP addresses from where a previous successful login took place. In other words, even after a number of unsuccessful logins from random locations, we would still allow login from an IP address from which a previous login was successful. Thus, if the legitimate user tries to login from home, or other location previously used, the lock out based on the random location does not apply. Again, note that here we assume that if the attacker has not compromised the user's machine. If he did, he would likely install a keylogger at once, and therefore would not need to try to brute force the password.

With the above lock out strategy, the user is very little inconvenienced, and unlikely to be locked out of the account. Yet, an attacker controlling a botnet of 10,000 machines, can only execute 250k login trials per year. If the attacker is in possession of the full list of userID's, and attacking an institution that uses only 6 digit pin, it would have a success probability of 25%. Nevertheless, increasing the userID space, such that only one in 1000 userID's are valid, the probability reduces to 0.025%. And the time necessary for the attack grows geometrically: an attacker would need another year to increase the probability of success to 0.026%.

5. RELATED WORK

An early study of password habits by Morris and Thompson [15] revealed that user's tended to choose weak and easily guessed passwords. This study was carried out in a Unix environment, where stealing access to the file of password hashes presented the potential of an off-line attack. A more recent study of web password habits by Florêncio and Herley showed that weak passwords are still very common, but also that average users generally juggle as many as seven passwords, and re-use them multiple times across several distinct accounts. A US Department of Defense report [3] documents recommended password practices that echo many of the points raised in the introduction. While many of the recommendations are sensible and necessary when an offline attack is available part of the findings of this paper is that the lessons are less applicable to online account passwords. The Center for Password Sanity [1] maintains references on the burden of passwords practices that users are asked to follow.

Much of the work on password attacks has focussed on off-line attacks. A famous early large scale password cracking attack in 1988 is analyzed by Seeley [16]. As with the majority of password cracking systems this system exploited access to files of hashed passwords. In [7] Di Crescenzo *et al.* propose password protocols which are both computationally efficient and provably secure in the sense that an offline attack is not stronger than an online attack. This is achieved by keeping password data in huge files which are almost impossible to be obtained by adversaries. In [12], Kedem and Ishihara propose a hardware based strategy to crack Unix passwords in reasonable time. A very powerful generic offline password cracker is JohnTheRipper [2].

In [17] , Yan et. al. conduct a user study on the mem-

orability and the security of the passwords. They suggest that the passwords should have random looking but mnemonic nature such as Pass Phrases in order to be both secure and memorable. In [14] Pinkas and Sander propose a password protocol which increases the cost of dictionary attacks by incorporating a simple challenge along with the passwords which can be responded easily by humans.

There has been little work on the attacks on password systems where there is no off-line attack. In an excellent review Hole *et al.* [11] describe attack scenarios on an online bank including the bulk-guessing attack of Section 2.2. They also draw attention to the fact that large institutions are more vulnerable than small ones for the same password strength.

Bortz *et al.* [5] shows that valid userID's can be discovered on-line using a a timing attack. Cheswick *et al.* [6] point out the importance of strong passwords where off-line attacks are possible, but also make clear that this offers no defence against password stealing spyware. Adams and Sasse [4] suggest that user's are not easily motivated by security threats they do not understand. This would add strength to our claim that addressing bulk guessing attacks (Threat 4) by increasing password strength is a poor approach. Many users may not understand the threat, and it is an attack on the institution rather than the user. The work of [4] suggests that placing the burden on users is undesireable, given that there is a simple alternative. Password stealing threats are addressed in [8, 13, 10]. It is probably true to say that these represent a bigger threat to online accounts than brute-force attacks.

6. CONCLUSION

We examine the question of attacks on password-protected web accounts. We conclude that forcing users to choose strong passwords appears misguided: this offers no defence against the common password stealing attacks and there are better means to address bulk guessing attacks. We show that it is the combined size of the userID plus password key-space rather than the password key-space alone that protects large institutions against bulk guessing attacks. Greater security for the institution can be achieved by allowing users to keep relatively short passwords, so long as they choose longer userID's. This reduces the number of break-ins that an attacker with fixed resources can expect, and reduces the burden on users. For smaller institutions, *i.e.* those with hundreds rather than millions of users, there appears to be little reason to use strong passwords so long as good lockout (*e.g.* three unsuccessful logins freezes the account for a time) are in place.

Note: the authors would like to stress that this paper is by no means intended as advice to end-users on password practices. Rather, in the spirit of the HotSec workshop, it is intended to provoke discussion on the rôle of password strength in securing web accounts.

7. REFERENCES

[1] `http://www.smat.us/sanity/`.

[2] `http://www.openwall.com/john/`.

[3] Department of Defense Password Management Guideline. Technical Report CSC-STD-002-85, U.S. Dept. of Defense, Computer Security Center, 1985.

[4] A. Adams and M. A. Sasse. Users are not the enemy. *Commun. ACM*, 42(12), 1999.

[5] A. Bortz, D. Boneh, and P. Nandy. Exposing Private Information by Timing Web Applications. *WWW 2007, Banff.*

[6] W. Cheswick, S.M.Bellovin, and A. Rubin. Firewalls and Internet Security. *Addison-Wesley*, 2003.

[7] G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In *Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA,*, pages 225–244, March 2006.

[8] Dinei Florêncio and Cormac Herley. How To Login From an Internet Café without Worrying about Keyloggers. *Symp. on Usable Privacy and Security*, 2006.

[9] D. Florêncio and C. Herley. A Large-Scale Study of Web Password Habits. *WWW 2007, Banff.*

[10] D. Florêncio and C. Herley. KLASSP: Entering Passwords on a Spyware Infected Machine. *ACSAC*, 2006.

[11] K. J. Hole and V. Moen and T. Tjostheim. Case Study: Online banking Security. *IEEE Security & Privacy Magazine*, 2006.

[12] G. Kedem and Y. Ishihara. Brute force attack on unix passwords with simd computer. In *Proceedings of the 8th USENIX Security Symposium, Washington, D.C.*, pages 93–98, August 1999.

[13] A. Pashalidis and C. J. Mitchell. Impostor: A single sign-on system for use from untrusted devices. *Proceedings of IEEE Globecom*, 2004.

[14] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 161–170, 2002.

[15] R. Morris and K. Thompson. Password Security: A Case History. *Comm. ACM*, 1979.

[16] D. Seeley. Password cracking: a game of wits. *Commun. ACM*, 32(6), 1989.

[17] J. Yan, A. Blackwell, R. Anderson, and A. Grant. The memorability and security of passwords – some empirical results, 2000.