nccgroup
freedom from doubt

An NCC Group Publication

“If your password is ‘password’, then it doesn’t matter how good your security is” or “Why password and brute-force mitigation policies matter”

Prepared by:

Will Alexander

Contents

1 Introduction

Passwords are the most prevalent method for providing users with the means to authenticate themselves in the digital world. Typically, all that is required to log in to a system is a user name, or other unique identifier, and a password – a simple string of characters that should only be known by the user.

Given the opportunity, users will often select weak (and therefore easy to guess) passwords, leaving their online identities vulnerable to compromise in the absence of other security controls. This assertion is borne out by many of the security assessments we perform[1]. It is common for a penetration tester to compromise a system by first gaining unauthorised access to a user account with a weak password – and the same is likely true for the real threat actors that penetration testers are simulating. Therefore, it is incredibly important that users (are forced to) select, or are issued with, passwords that make it difficult for others to impersonate them, and that adequate measures are in place to prevent attackers from even attempting to do so.

In this whitepaper we discuss the need for good password and brute-force mitigation (or account lockout) policies[2], for both operating systems and web applications, to help minimise the likelihood of user accounts being compromised.

We use RFC-style key words to indicate policy requirement levels[3]. It will become apparent that many requirements are "recommended", meaning that there may be valid reasons to ignore a particular requirement. For example, some password policy requirements might be ignored, for the sake of usability, if for example the brute-force mitigation policy is sufficient to prevent attackers from easily guessing passwords. However, the full implications must be understood and carefully weighed before choosing a different course. For example, an organisation might choose not to enforce a "strong" password policy for the sake of usability, with the understanding that the passwords selected by users might not be as secure as they could be and are "strong enough" to protect the information or functionality it protects.

[1] A discussion of how weak password and account lockout policies can be abused to gain unauthorised access to user accounts can be found at https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2013/july/making-a-mountain-out-of-a-mole-hill-combining-low-severity-vulnerabilities-to-devastating-effect/.

[2] It's important to note that user accounts, and thus password and brute-force mitigation policies, should only be deployed where they are strictly required. If authorisation is not required to access a resource, then user accounts are likely not necessary.

[3] https://tools.ietf.org/html/rfc2119

2 Password Policy

The aim of a password policy is to ensure that users select passwords that will not easily be guessed by an attacker – assuming no other compensating controls, such as an adequate brute-force mitigation policy.

2.1 Passwords must not be less than a specified minimum length

Passwords must be greater than a specified minimum length because short passwords are generally easier to guess.

To demonstrate this, imagine someone (a user) picking a number between 0 and 9 (a password exactly one character in length containing only digits), and asking another person (an attacker) to guess the number they have picked. It's not difficult to imagine that the attacker will be able to guess the number pretty quickly. Next, imagine picking a credit card PIN between 0000 and 9999 (a password exactly four characters in length containing only digits) and repeating the process[4]. It would probably take them a little longer, but they'd likely guess the PIN within a reasonable amount of time. Now imagine scaling this up to asking someone to guess a PIN with values between 00000000 and 99999999[5] (a password exactly eight characters in length containing only digits). There are 100 million possible numbers that could have been chosen, so it will very likely take the other person a while to guess the correct number.

Typically, a minimum password length of eight characters is recommended[6], but longer is generally better. Also, the minimum length should be increased as the value of the user account increases. For example, passwords for users with some kind of administrative function should be longer than those for standard users.

Passwords are usually six to ten characters in length as they are typically based on words. A passphrase, which is typically based on a phrase, is similar but much longer – twenty to thirty characters is typical – making brute-force password guessing much more difficult. The use of passphrases by users is highly recommended to increase their online security. However, to ensure that passphrases are not easily guessed they should not be based on a phrase in a book of quotations or phrase compilations. This whitepaper uses the term password throughout for the sake of simplicity and clarity, but in all instances the term passphrase could be used instead.

Note that by specifying a minimum password length, the total number of possible passwords is marginally reduced. Specifying a minimum password length (obviously) does not allow for passwords that are less than that minimum length. Using the previous example, specifying a minimum password length of eight digits does not allow passwords that are between zero[7] and seven digits in length, thus eliminating 11,111,111[8] possible passwords. This is a reasonable trade-off to prevent users from selecting short (and thus easily-guessed) passwords.

2.1.1 Password lengths should not be constrained to an upper limit

There should be no upper limit on the length of passwords, because such a limit decreases the total number of guesses that an attacker must make when guessing passwords.

The previous scenario forced the user to choose a number made up of exactly eight digits, which, as already mentioned, limits the total possible number of guesses that the attacker would need to make to guarantee success. Now assume that there is no upper limit on the range of numbers that the user

[4] An analysis of credit card PINs was performed, and the results can be found at http://www.datagenetics.com/blog/september32012/.
[5] Imagine looking at the odometer on the dashboard of a car.
[6] As recommended by Microsoft at https://technet.microsoft.com/en-us/library/hh994560.aspx.
[7] A password made up of 0 characters would be an empty password, which is a valid (albeit not very good) value.
[8] $10^0+10^1+10^2+10^3+10^4+10^5+10^6+10^7 = 11,111,111$

can pick – that is, they can pick any number between “00,000,000” and infinity! Given this range of possibilities it is practically impossible that the attacker will be able to attempt every possible number, and very unlikely that they will be able to guess the correct value, if the user has picked a sufficiently large value.

The advice to implementers is to not limit the length of passwords that users can select. Doing so severely reduces the maximum number of guesses that an attacker would need to make before correctly guessing the password. Any upper limit on the character length is a significant reduction from infinity. Mechanisms (namely hashing (not discussed in this whitepaper)) exist to store representations of passwords of any length, so there is no reason to place an upper limit on the length of passwords users can select.

There is, of course, an upper limit to the length of passwords that computers can handle, but this value is so high that users are never going to realistically select a password with that number of characters.

2.2 Passwords should contain a number of different character classes

Passwords should contain a number of different classes, which can be defined as:

- **Upper-case letters,**
- **Lower-case letters,**
- **Digits, or**
- **Special characters (i.e. a character that is not in one of the aforementioned groups, such as ‘£’, ‘$’ or ‘%’).**

The more character classes used in a password, the larger the number of guesses that an attacker must make when brute-forcing passwords.

When discussing password length, we imagined a scenario where the attacker had to guess a number picked by the user. The only characters used were digits, and as such there were only ten possible values each character could take. Each digit appended to the number increased the total number of possible guesses by a factor of ten. In a more general sense, the total number of possible guesses the attacker would need to make to be sure to correctly guess the user’s password could be calculated as:

Number of possible values for each character raised to the power of the length.

Let’s amend the rules of the first “game” by saying that the user can pick any number between 0 and 9 or a lower case letter – a password - and the attacker has to guess what was picked. The attacker could make a maximum of thirty-six possible guesses before guessing correctly, because the user could have selected any of the twenty-six lower case letters ‘a’ through ‘z’ or any of the ten digits ‘0’ through ‘9’. Now consider that the length of the string the attacker has to guess is exactly two characters in length and made up of this thirty-six-character set. The first character could take one of thirty-six values, as could the second. Therefore, there are a maximum of 1,296 guesses the attacker could make before guessing correctly. Expanding the length of the string to exactly eight characters makes the maximum number of possible guesses 2,821,109,907,456 (36 ^ 8) – approximately 2.8 trillion.

Including upper-case letters as well increases the total number of characters that the user can pick to sixty-two, so a string of exactly eight characters would take the attacker a maximum of 218,340,105,584,896 (62 ^ 8) –approximately 218 trillion - guesses.

If some special characters (for example, the top ten characters along the top of a British keyboard: ‘!’, ‘“’, ‘£’, ‘$’, ‘%’, ‘^’, ‘&’, ‘*’, ‘(‘ and ‘)’) are also added then the total number of characters that the user

can pick is 72[9], and a string of exactly eight characters would take the attacker a maximum of 722,204,136,308,736 (72 ^ 8) – approximately 722 trillion – guesses[10].

Note that the above examples assume that users are forced to select passwords exactly eight characters in length. As already mentioned, mandating an upper limit is not considered good practice as it limits users' choices. Therefore, 722 trillion guesses are just the total number of guesses required to ensure that all possible eight-character passwords have been attempted. An attacker would then need to attempt all possible nine-character passwords, and so on.

Also note that the maximum number of actual attempts required to guess a password exactly eight characters in length is actually lower than 722 trillion, since the choices have been restricted by stating that the password must contain at least three of the four defined character classes. For example, passwords such as "12345678", "abcdefgh", and "ZYXWVUTS" are not valid choices because they do not contain at least three of the defined character classes. However, this reduction in possible passwords is considered an acceptable trade-off, as it forces the user to select a password that is more difficult to guess.

Typically, requiring users to have at least three of the four main character classes in their password is sufficient to create a password that is not trivial for an attacker to guess[11].

2.3 Passwords should not be based on common dictionary words

Passwords should not be based on common words alone, to avoid an attacker easily guessing passwords.

The previous example assumed that the user was equally likely to pick "KJ6E&jBd" as they were "P@ssw0rd". Statistically this is true, but users are generally not statistically random in their choice of password – especially if they are required to remember it without writing it down. Due to the need for users to remember so many passwords they are much more likely to choose "P@ssw0rd" than they are "KJ6E&jBd".

Attackers guessing passwords will exploit this weakness by guessing them early on. This is known as a dictionary attack[12], and is commonly performed early on when guessing passwords.

Attackers will also perform what is known as a hybrid attack, which takes a dictionary of words and creates new variations based on rules. For example, a rule could specify that every word in the dictionary be appended with a number, or another might specify that the every word in the dictionary be converted to upper case. Rules might also convert words in the dictionary to "l33tspeak", where letters are converted to similar looking numbers. For example, 'e' would become '3' and 'o' would become '0'. Using a hybrid approach, attackers can guess passwords that are variations of those already in their dictionary.

Attackers will then commonly launch password attacks based on machine learning techniques and then password masks before embarking on a true brute-force guessing attack where they try every other possible password not already tried as part of an earlier attack[13].

---

[9] The actual total number of characters available on a British keyboard is 95 so the maximum number of guesses would be much greater.

[10] The maximum number of guesses required by an attacker based on password length and the password complexity can be computed using the online "Brute Force Password Search Space Calculator" at https://www.grc.com/haystack.htm.

[11] [11] As recommended by Microsoft at https://technet.microsoft.com/en-us/library/hh994562.aspx.

[12] There are a number of widely available dictionaries that attackers use based on previous breaches. The most commonly used is the "rockyou" dictionary available from Skull Security at https://wiki.skullsecurity.org/Passwords. A skilled attacker will combine a number of dictionaries.

[13] A breakdown of a typical password attack can be found at https://www.praetorian.com/blog/statistics-will-crack-your-password-mask-structure.

As well as common words, passwords should not be based on the username or the name of the company or service to which the user account provides access, as these are commonly guessed by attackers early on any password attack.

Note that by specifying that certain passwords are not permitted the total number of possible passwords that a user can select is reduced. However, this reduction in possible passwords is considered an acceptable trade-off, as it forces the user to select a password that is more difficult to guess.

2.4 Passwords should be unique (especially on high value accounts)

Passwords should be unique, to prevent the compromise of other passwords.

As previously mentioned, the attacker uses dictionaries to exploit the statistical bias in users' password selections. Any compromised passwords not already in their dictionary will be added for use in future attacks.

For users this means that the passwords they select should be unique for each account they have. No two user accounts should share the same password, since an attacker able to compromise one of your passwords will likely use that same password in an attempt to gain unauthorised access to other accounts owned by them. User accounts owned by the same user will likely be easy to identify since the username will often be the same as the user's email address.

Users might consider the use of a password manager, such as LastPass[14] or KeePass[15], to manage their passwords, allowing every user account to have a complex, long and unique password. However, such software is not impregnable and is likely to be a target for attackers. The master password used to manage this software should be as long, complex, and random as possible to prevent it being easily guessed by an attacker.

However, research published by Microsoft[16] suggests a more nuanced approach. The research suggests users should use and reuse easy to remember (thus weak) passwords for websites which don't hold valuable information, and instead focus on remembering complex and unique passwords for high-value sites, such as banking or e-commerce. Where passwords are reused their compromise must not result in the attacker gaining access to more sensitive information using the same password. For example, the same password must not be used to gain access to a message board as is used to gain access to a banking application. There is some merit to this approach, as the paper states that remembering complex passwords for a hundred user accounts would be equivalent to remembering pi to 1,361 places, which is very difficult for the majority of users.

For administrators, ensuring passwords are not reused means that when new user accounts are created or when existing user accounts' passwords are reset it is important to select random and unique passwords for each user. Do not use a common password such as "Welcome1", since any compromised password is likely to be added to an attacker's custom dictionary and used in future dictionary attacks. If all new users are given the same password and one is compromised, then this could lead to the compromise of all other accounts with the same password.

2.5 A mechanism for changing passwords must be provided

Users must be able to change their passwords on demand in the event that they believe their password has been compromised.

Users should be able to elect to change their password whenever they feel appropriate to do so. For example, if a user has reason to believe that their password has been compromised, then they must be provided a mechanism for changing it.

[14] https://lastpass.com
[15] http://keepass.info
[16] http://research.microsoft.com/pubs/217510/passwordPortfolios.pdf

Allowing users to regularly change their password also decreases the window of opportunity for attackers to successfully guess passwords.

2.5.1 The mechanism for changing passwords must require the current password

Users must provide their current password when changing passwords, to prevent an attacker changing it.

When a user changes their password through normal functionality (i.e. when the user is not changing their password because they have forgotten it) they must be required to provide their current password. If the current password is not required, then it might be possible for an attacker who has hijacked a legitimate user's session[17] to take over the hijacked user account by changing the password to a value of their choosing.

2.6 A maximum password age may be used

A maximum password age may be set, to force users to regularly change their passwords, to decrease the window of opportunity to successfully guess passwords.

Not all users will change their password of their own volition, and the odds are in the attacker's favour that they will guess a user's password eventually – especially if the user picked a password that is relatively easy to remember (and thus easy to guess by an attacker). Therefore, it is recommended that the user be forced to regularly change their password – every ninety days is typical. This decreases the window of opportunity for the attacker to correctly guess the password, because after ninety days, for example, the password will have changed so the attacker will have to start their password attack from the beginning.

Note though that forcing users to regularly change their passwords may actually decrease security. It's already been discussed that users should have unique passwords for each system. Forcing them to remember new passwords for each system every ninety days further increases the burden on the user. A better strategy may be to only force users to change their password when there is an indication that their password has been compromised.

2.6.1 A password history should be maintained, if a maximum password age is used

A history of passwords should be maintained to ensure that the same password is not used when the password is changed.

When forced to change their passwords, users may try to change their password to the same value (i.e. not change it all), or select a password from a small pool of favourites that are easy to remember or which they believe to be secure. Unfortunately, passwords can be compromised and an attacker with suitable knowledge about the user could abuse their small pool of passwords to gain unauthorised access to their account(s).

To ensure that users do not choose a password that they have recently used, a password history should be maintained. Users should then be prevented from selecting a password that appears in their password history. This prevents users from selecting passwords from a small pool of recycled choices. A password history of twenty-four passwords remembered is typical[18].

[17] A user's session could be hijacked using any number of means, from a user simply leaving their unattended workstation unlocked to a cross-site scripting exploit (in the specific case of web applications).

[18] As recommended by Microsoft at https://technet.microsoft.com/en-us/library/hh994571.aspx.

2.6.2 Repeated password changes should not be permitted within a short period of time, if maintaining a password history

A minimum password age should be set to prevent users from repeatedly changing their password within a short period of time to bypass the restrictions placed upon them by a password history.

If a password history is maintained, then it is important that users are not permitted to repeatedly change their password over a short period of time. A determined user who wants to keep the same password could repeatedly change their password until their password history no longer contained their original password. They would then be free to select their original password. Therefore, if a meaningful password history is to be maintained, then it is important that users not be allowed to change their passwords immediately. Typically, a minimum password age of one day is long enough to prevent users from cycling through a number of passwords before returning to their original.

2.7 Passwords must be stored appropriately

Passwords must be stored appropriately, such that, in the event of their disclosure to an attacker, they cannot be easily used to gain access to user accounts.

This paper will not cover the specifics of how passwords should be stored[19], but it is important to discuss the topic in terms of password policies and how it affects the passwords that users should select.

Passwords must not be stored in plaintext, so that in the event that the system on which user information is stored is breached, attackers are unable to easily obtain users' passwords. Similarly, passwords must not be encrypted. If passwords are encrypted, then the key must be stored. An attacker who has been able to obtain the encrypted passwords will also likely be able to retrieve the key used to encrypt them.

It is recommended that password hashes are stored. Hashes are password representations from which it is difficult to derive the original password. Up until now we have discussed guessing passwords in the context of an online attack where an attacker attempts to directly gain access to the system using guessed passwords[20]. An attacker in possession of users' password hashes would perform an offline password attack to "crack" the hashes to retrieve the plaintext passwords that can be then be used to gain access to the system. An offline password attack can be performed many magnitudes quicker than an online attack, so it is important to ensure that password hashes are properly stored and protected using unique salts for each user and multiple hashing iterations.

2.7.1 Passwords must never be transmitted in clear text

Passwords must never be transmitted in clear text, such as via email, because an attacker able to intercept traffic destined for the intended user would be able to retrieve it.

Rather than sending web application users their passwords, an email hyperlink could be sent to users' registered email addresses. When accessed, the link takes the user to a secure part of the application to set their password. The hyperlink:

[19] A discussion on password storage is available on the NCC Group blog at https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2015/march/enough-with-the-salts-updates-on-secure-password-schemes/.

[20] An attacker would also need valid usernames against which a password attack could be launched. A discussion on username enumeration is available on the NCC Group blog at https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2015/june/username-enumeration-techniques-and-their-value/.

- Must contain a cryptographically strong token to uniquely identify the user,
- Must be usable only once, and
- Must expire after a short period of time.

For increased security, a security token, or out-of-band value (such as a one-time PIN transmitted via SMS), could be used in conjunction with the emailed hyperlink to reduce the threat of an attacker intercepting the email and using the reset hyperlink before the legitimate user had chance to do so.

The same mechanism could be used to allow users to reset their password if it is forgotten. When a user requests a new password via the web application's forgotten password functionality they would be sent the aforementioned hyperlink[21].

Such a system would be unlikely to work for communicating passwords for operating system user accounts, since it is unlikely that a user will have access to their email without first gaining access to their workstation. Instead passwords could be transmitted via telephone or SMS. In these instances, passwords must be reset after their first use to ensure that system administrators do not know users' passwords.

2.8 Multi-factor authentication should be used

Using multi-factor authentication can prevent the compromise of users' accounts in the event that their password is compromised.

When a user authenticates using just a username and password, they are using a knowledge factor ("something only the user **knows**") to gain access. That is, they are using just a secret string of characters to prove that they are who they say they are. In the event that a user's password is compromised, an attacker would very likely be able to gain unauthorised access to their user account. To protect against this scenario, multifactor authentication can be used.

As well as requiring a user to provide a knowledge factor, systems can specify that users provide a possession or inherence factor:

- A possession factor ("something only the user **has**"), in the context of physical security, could take the form of locks and keys. That is, someone wanting to gain access to a building would typically need to have a key that will fit the protecting lock. In the context of information security, a possession factor would likely be a security token. Security tokens can take many forms, but the simplest have a physical display, which regularly changes, and the user is required to provide the currently displayed value. These tokens could be hardware-based, such as RSA SecurID[22] or software-based, such as Google Authenticator[23].
- An inherence factor ("something only the user **is**") is typically biometric, meaning that the user authenticates using their finger, retina or voice. Modern Apple iPhones allow users to authenticate using their fingerprint using Touch ID[24].

Multi-factor authentication can drastically reduce the compromise of users' accounts, since it requires more than just the user's password to gain access. It is often recommended that multi-factor authentication be used[25] – particularly on Internet-facing systems. However, it is not without its problems, as it does remain vulnerable to man-in-the-middle and Trojan attacks[26].

[21] OWASP provides a generic framework solution for implementing self-service password resets at https://www.owasp.org/index.php/OWASP_Periodic_Table_of_Vulnerabilities_-_Insufficient_Password_Recovery.
[22] https://en.wikipedia.org/wiki/RSA_SecurID
[23] https://en.wikipedia.org/wiki/Google_Authenticator
[24] https://en.wikipedia.org/wiki/Touch_ID
[25] https://www.usenix.org/system/files/conference/soups2015/soups15-paper-ion.pdf
[26] Bruce Schneier discusses these attack vectors on his blog (Schneier on Security) at https://www.schneier.com/blog/archives/2005/03/the_failure_of.html.

3 Brute-Force Mitigation Policies

While a good password policy will help users protect their accounts, a determined and skilled attacker could still gain unauthorised access. The aim of a brute-force mitigation (or account lockout) policy is to limit the chances of an attacker doing this even when users have selected poor passwords[27].

3.1 Accounts should be made inactive after a number of consecutive failed authentication attempts

There should be a penalty for making multiple consecutive failed authentication attempts, to limit the success rate of online password guessing attacks.

To limit the chances of an attacker successfully guessing users' passwords as part of an online brute-force attack, user accounts should be made inactive (or "locked out") after a number of consecutive failed authentication attempts. A good password policy will prevent users from selecting passwords that are likely to be guessed before user accounts are made inactive by the account lockout policy.

The precise number of consecutive failed authentication attempts permitted will depend on the trade-off that can be made between usability and security. Users will often mistype or forget their passwords so some leniency should be allowed. However, the more failed authentication attempts that are permitted, the more likely an attacker is to successfully guess a user's password. Typically, between four and ten consecutive failed authentication attempts are permitted before a user account becomes inactive[28].

3.1.1 A failed authentication counter may be used

A counter may be used to track the amount of time elapsed since the last failed authentication attempt. When the counter reaches a set value the number of logged failed authentication attempts is reset to zero.

To allow users to perform new authentication attempts after a number of failed logons without locking their account, a counter can be used to track the amount of time that has elapsed since the last failed logon. When the counter reaches a defined value, the number of logged failed authentication attempts is reset to zero, allowing the user to make more authentication attempts without fear of making their account inactive.

If the counter value is set too high, then users may lock themselves out for an inconveniently long period of time. If this value is too low, then it increases the likelihood of an online password attack succeeding. A reasonable value allows users to perform new authentication attempts after a number of consecutive failed authentication attempts, without making online password attacks feasible at high speeds.

It should be added that this feature would do nothing to increase security. Instead, it could be added to an account lockout policy as a concession to the idea that there can be a cost associated with making user accounts inactive.

[27] OWASP has a resource on how to prevent brute-force attacks from succeeding at https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks.
[28] As recommended by Microsoft at https://technet.microsoft.com/en-us/library/hh994574.aspx.

3.1.2 Account lockouts must apply to all stages of authentication

If an authentication mechanism has multiple stages, then the brute-force mitigation policy must be applied to all stages.

To increase security, some organisations may require users to authenticate over a number of stages or use multiple factors. For example, users might be required to provide a username and password, followed by a number of characters from a secret word. The reasoning behind this is that a user will require two pieces of information to successfully authenticate: the password and the secret word. Note that this example is not multi-factor authentication, as both stages require something the user knows.

If an attacker is able to compromise the password but not the memorable word, then they will need to perform a brute-force guessing attack against the characters requested from the memorable word. To assist in preventing unauthorised access to user accounts, brute-force mitigation policies should be equally applied to all stages of authentication.

3.1.3 Accounts should be made inactive for a limited period of time

User accounts should not be made inactive indefinitely, to avoid placing an unreasonable burden on administrators.

Making accounts inactive indefinitely until manually unlocked by an administrator might be preferable, but could be exploited by an attacker to perform a denial-of-service of sorts by purposely locking a large number of user accounts, which would then need to be manually unlocked by an administrator.

Instead user accounts should be made inactive for a period of time before becoming active again. This does increase the risk of an attacker successfully guessing a user's password by performing an online password attack, but reduces the risk of large numbers of user accounts being locked out indefinitely, though these risks may vary depending on the system being accessed.

A decision is needed to balance the risks of an online password attack succeeding and users' accounts being unavailable in the event that they are locked. Locking accounts for thirty minutes is typical[29], but may vary depending on the system being accessed.

It is important to ensure that a strong password policy is in place if accounts are not locked indefinitely after a number of consecutive failed authentication attempts, to prevent successful online password attacks, and that adequate monitoring is in place to detect such attacks.

3.2 Mitigations must be in place to prevent attackers gaining access to user accounts

If an account lockout policy is not appropriate for the environment, then other mitigations must be in place to prevent an attacker from gaining unauthorised access to user accounts.

Making user accounts inactive after a number of failed authentication attempts is not always appropriate. For example, consider a web application that offers online auctions. Several users could be bidding on the same item. If this application enforced an account lockout policy, then one user could lock the others' accounts in the last moments of the auction, thus preventing them from submitting any winning bids. The same technique could be used to block other time-sensitive applications, such as those offering financial services.

One useful mitigation for web applications might be the introduction of a CAPTCHA after a number of failed consecutive authentication attempts before making user accounts inactive. This would limit the ability of an attacker to automatically lock user accounts, as it would require some interaction from a real user to solve the challenge posed by the CAPTCHA.

[29] As recommended by Microsoft at https://technet.microsoft.com/en-us/library/hh994569.aspx.

Another mitigation for web applications might be to prevent users from logging in from unknown devices or IP addresses. An attempt to log in from a new device or IP address would require the user to verify that it is them attempting to access the service by clicking on a secure hyperlink sent to their registered email address.

4 Conclusions

Passwords are difficult for both users and organisations to manage and an alternative is urgently needed. Passwords can be bolstered with the use of multi-factor authentication, but good password and brute-force mitigation policies are crucial in preventing the compromise of user accounts. However, defining these can be nebulous and will depend on a number of factors, including cost and usability. However, some broad recommendations can be made with the specifics varying between organisations, as well as individuals.

Password policies:

- Must mandate that passwords are greater than a specified minimum length.
- Should specify no upper limit on the length of passwords.
- Should specify that passwords contain at least three of the four main character classes.
- Should specify that passwords are not based on dictionary words, usernames, or anything relating to the system to which the user account provides access.
- Should specify that passwords are unique to each user.
- Must mandate that a mechanism is in place to allow users to change their password on demand.
  - Must mandate that this mechanism requires the current password.
- May specify that passwords are changed regularly.
  - Should specify that a password history be maintained, if users are regularly forced to change their passwords.
    - Should prevent passwords being repeatedly changed in a short period of time, if a password history is maintained.
- Must specify a secure means to store and transmit passwords.
- Should specify that multi-factor authentication be used.

Brute-force mitigation policies:

- Should specify that user accounts are made temporarily inactive after a number of consecutive failed authentication attempts.
  - Must apply to all stages of authentication.
  - Should only make user accounts inactive for a period of time, or
- Should specify other mitigations where making accounts inactive is not appropriate, such as the use of a CAPTCHA or requiring user interaction if an authentication attempt is made from an unknown location or device.

5 Acknowledgements

The author wishes to thank Ed Williams, Jerome Smith and Ollie Whitehouse of NCC Group for their peer review and valued suggestions.

