Better Passwords and Policies

David R. Patten

East Carolina University

Abstract

Everyday press releases cover information security breaches from many top companies. Often these breaches include releasing the passwords that were stolen. Sadly, these passwords are usually trivial to crack which leads to further exploitation. Simple passwords are a problem that is created by both human nature and by poor password policies and training. This paper will look at the tools used to crack passwords, the passwords users create, password policies, and look at some creative and innovate solutions to the password problem.

Keywords: Passwords, Hash, Security Policy, Password Policy, Information Security

Better Passwords and Policies

In December 2009, 32 million account passwords were stolen from RockYou (Takahashi, 2009). It was an unprecedented breach in security that exposed millions of user passwords, and revealed how bad users are at picking secure passwords. Today, the news is littered with stories of password breaches that expose user's passwords. Often these password dumps are sold on the black market and the passwords are used to compromise other accounts. This paper will examine password cracking tools and techniques, look at the passwords that users create, examine some password policies, proper password creation methods, and some recommendation in the conclusion.

Password Cracking Tools

The news is littered with various reports of password breaches. Often these breaches occur from vulnerabilities being exploited on an organizations server. After the exploitation, the hacker will often extract the user's account information. Generally, passwords are stored using a one way encryption algorithm. The result of the encryption process is called a hash. Once the hacker has the password hashes, the attacker must decipher the hashes to discover the password. The process of deciphering the password is called "cracking".

Various methods are used to crack a password from a hash. Since the encryption algorithms are one way, meaning the hash cannot be reversed, the methods involve testing various passwords against the hash until the password creates a hash that is identical to the hash that was stolen. One method of password cracking is using rainbow tables. Rainbow tables contain every possible hash for a given algorithm and specific criteria. (Documentation, n.d.) Rainbow tables can be freely downloaded online. The Tables are categorized usually by length, criteria, and algorithm. The length is the length of the password the table is built for. The criteria are the characters used in the table such as upper case letters, lower case letters, numbers, and symbols. Usually rainbow tables are only available for the most common algorithms. Rainbow tables are effective against passwords that meet the criteria for the table. For example if a password is ten characters consisting of upper and lower case letters, a rainbow table that is created for both upper and lower case letter with a password length of ten in the correct algorithm will crack the password. There are some limits with rainbow tables. In the previous example, a rainbow table designed for both upper and lower case letters with a length of nine characters would not crack the ten letter password. Rainbow tables are very specific and only work with passwords that meet the criteria for which the table was designed. Another limitation of rainbow tables is size. A rainbow table for passwords that are nine letters in length can be hundreds of gigabytes. Also, most rainbow tables are limited to ten or less characters and many do not include symbols. These limitations have lead hackers to use other methods for cracking passwords.

Programs have been created that are designed to crack passwords by trying words in dictionaries, combinations of letters, number, and symbols, and combinations of words, letters, numbers, and symbols. John the Ripper or Hashcat are examples of password cracking programs. A password cracker has various ways to attack a password hash.

A common attack used is a dictionary attack. A dictionary attack uses a wordlist, or dictionary. The password cracker calculates the hash to each of the words in the wordlist and compares them to the list of hashes. The dictionary attack is only as good as the dictionary used. Many hackers have specialized dictionaries that consist of passwords that have been previously cracked and collections from some of the largest password dumps.

Another common attack is called a brute force attack. The brute force attack will try every combination of letters, numbers, and symbols starting with 1 character and continuing to add characters, while trying every permutation, until the password is cracked. (Hashcat Wiki, n.d.) Brute forcing a password has some limitations. For example, modern hardware can brute force an eight character password in a couple hours but as the password length increases the time increases exponentially. Longer passwords could take years to crack. Brute forcing a long password can be resource intensive and at times impractical. As a result, password crackers have developed more complex methods for cracking password hashes.

The mask attack is an advanced password cracking technique that is available to attackers (Hashcat Wiki, n.d.). This technique combines the dictionary attack and a brute force attack by either appending or prepending characters to the front or back of a word in the dictionary. This technique is useful for cracking longer and more complex passwords. For example if the password is “password123”, a dictionary that does not contain password123 would not crack the password. Also it would take a long time to use the brute force method because the password is eleven characters in length. However, a mask attack would crack this password quickly. The attack would take the base word “password” and add numbers to the end of the base word. The mask attack is defined by the user. Generally the type of characters added to the base word is defined by the user. The user has the option of adding numbers, lower case letters, upper case letters, upper and lower case letters, symbols, or all characters. Using these options, the password cracker can tailor the attack to the requirements of the password. If an organization requires a number to be part of the password, the mask attack can add a numbers to all the words in the dictionary. By tailoring the attack to the organization, the attacker will crack more passwords in a shorter time.

A rule based attack is an attack that manipulates the words in the dictionary to allow flexibility that the standard dictionary attack does not provide (John the Password Ripper, 2015). The rule based attack can add characters like the combination attack, but has other capabilities as well. The rule based attack allows the password cracker to accept a simple programming language. The rules are either placed in rule files or added options to the program. These rules give the ability to script what manipulations are made to the words in the dictionary file. Manipulations such as making the word all upper case, adding numbers to the end of the word, or making the first letter upper case. These manipulations allow the password cracker to crack more passwords. Password crackers that use this method will have some standard rule files that can be used for rule based attacks. In addition, elite password crackers have released rule files that have their most successful rules. In addition to using the rules, password cracking programs can allow a user to use rules with the mask attack.

Markov attack, named after the mathematical model Markov chain, is similar a brute force attack but takes into account the statistical analysis of the previous cracked passwords. (Turner, 2012) The statistics of previously cracked passwords are stored in a table. The table is then used to calculate the probability of the characters in a password. For example instead of incrementing through all the letters in the alphabet after the letter “q” a Markov attack would select the letter “u” which is more likely to follow “q”. In a large list of password hashes, the Markov attack can be a valuable tool to cracking the longer, more difficult passwords that other techniques fail to crack. This attack is especially effective on larger password dumps as the statistics gathered from the half cracked passwords may give insight into the remaining hashes. This method is built in to tools such as John the Ripper. Other tools, such as hashcat, use secondary tools to analyze the statistics of the cracked passwords. Either way, the Markov attack is easy to setup and execute.

User Created Passwords

With all the tools designed to crack passwords, users need to create strong passwords. The problem is people are notoriously bad at creating complex passwords. An analysis of some previous password leaks indicates that the majority of passwords are very weak. A recent study from Xi'an Jiaotong University and Tsinghua University looked at over 6 million leaked passwords and provided some analysis of user passwords. According to the research, 45% of passwords were numerical only (Chao Shen, 2016). This was a surprising discovery considering that previous studies had indicated a user preference of alpha only passwords. The trend towards numeric only passwords is concerning as those passwords are weak. Another interesting result from the study was that 15% of the passwords can be found on the top 20 most common password list (Chao Shen, 2016). The most common password list includes passwords such as "123456789" or "password". Unfortunately, this indicates that 15% of users either do not care about the security of their password or do not have sufficient information to create a stronger password. While much of the research has pointed towards insecure and poor passwords, there are some encouraging trends. In the recent studies, the average password length was 9.46 characters with a staggering 97.83% of passwords having eight or more characters (Chao Shen, 2016). It is encouraging that users are seeing the need for longer passwords. Also researchers have found an increase in completely random passwords. An increase in completely random passwords is encouraging as completely random passwords provide the most security. Another study of 70 million passwords confirm the trend that users are beginning to create more complex passwords (Bonneau, 2012). This study found some interesting trends. One trend was that users who had changed their password at least five times had stronger passwords. This would indicate that the security minded user sees the importance of changing their passwords and creating passwords that are stronger than average. Sadly the study found that passwords were largely equally weak across all demographics (Bonneau, 2012) . The demographics included male, female, all age groups and several nationalities. The problem of bad passwords is a problem that spans the globe.

Password Policies

Password policies are how companies attempt to set a standard for passwords in their organization. A password policy is a vital part of any security policy. Password policies address the details of password management such as password length, password history, minimum password age, maximum password age, password complexity, and how the passwords are stored (Stanek, n.d.).

Password history policy regulates how often a user is able to repeat a password. Windows Server 2008 will allow a password to repeat every 24 times. The password history enforcement is often combined with the password minimum age to prevent users from changing the password multiple times quickly enabling them to use the same password after changing 23 times in a row. The idea behind this policy is that it will discourage users from alternating between several common passwords.

Minimum and maximum password age regulate how long a password is valid. While the minimum password age prevents rapid password changes, a maximum password age forces users to change their password. Many organizations have a maximum password age of 90 days. According to Microsoft best practices, higher security accounts should have the password changed every 30 to 90 days (Stanek, n.d.). The goal of password age is to force users to change passwords regularly to prevent an attacker from using comprised credentials in perpetuity.

Password length and complexity policies govern the requirements for password length and the types of characters required. On many systems the default for password length is zero (Stanek, n.d.). Obviously, the password length should be set to a more sane policy. For secure applications the password minimum length should be 14 characters. Password complexity can involve more than just the types of characters in the password. For example, the password complexity may set a rule such as the username cannot be part of the password (Stanek, n.d.). The classic password complexity policy is requiring upper and lower case letters, numbers and symbols. By forcing the user to use the entire key space, the policy increase the entropy of the password and makes the password harder to brute force.

The finalpassword policy involves storage of the passwords. The passwords should never be stored in clear text. Passwords should be stored in a hash that has been salted. A hash is a one-way cryptographic one-way function that takes an input and provides a reliable encrypted output, or hash. A salt is a string added to each password that is unique to the each user account (Richard Shay, 2016) Salting prevents users who have the same password from having the same hash. Salting the hash makes the cracking process harder because the cracker must discover the salt and the password in order to crack the hash.

All these password policies aid in creating a secure environment both by helping users create more secure passwords and securely storing the passwords. Password policies should be reviewed and updated as technology changes. For example, the password length requirements have changed as the ability to brute force longer passwords with new technology has occurred.

By reviewing the password policy periodically, an organization can adjust the password policies to take advantage of new security measures.

User Training

While password policies are important, training users on how to create and store passwords is critical to maintaining a secure environment. Users should be trained on how to create sufficiently secure passwords. There are several methodologies for creating strong passwords. The Bruce Schneier's Method, Electrum Method, and PAO Method are all good methods for developing strong passwords.

The Bruce Schneier's Method takes a phrase or sentence and uses either the first letter of word or substitutes for words to create long complex passwords that are easy to remember (Schneier, 2014). For example, take the sentence "I ate spam and eggs for breakfast at Granny's house over the summer!" and convert it to the password "I8s&efbaGhots!" This password would not be in any hacker's dictionary and would take years to brute force; however, it would be easy for the user to remember.

The Electrum Method, as made famous by the xkcd comic, combines random words to create password phrases (Lee, 2014). An important point to remember when using the electrum method, is that the words must be truly random. Using famous phrases such as "To be or not to be" defeats the purpose and makes it easier for an attacker to crack. Using truly random phrases such as "Dog sharp any paint" as a password is very secure as the password is long and will not be in any dictionary or literary work.

The PAO method creates an image in your mind that helps you remember your password. PAO stands for Person, Action, and Object (Lee, 2014). The idea is to create a sentence using a person doing an action with an object. For example, "Michael Jordan crawling over lava.", creates a strong image that is easy to remember. The password does not have to be the whole sentence either. This method advocates combining the words to create the password. The above example could be combined to create "MicJcrawlingovlava."

Another method of password generation that is increasing in popularity is using a password manager such as Lastpass or KeePass (Wawro, 2011). The advantage of using a password manager is that a user can create one strong password to access all the other passwords that are encrypted and stored inside the password manager. These password managers have the ability to create passwords that are completely random, and since the password manager stores the passwords, a user can quickly create random passwords for each account they have. A password manager increases the security for the user by allowing the user to have a unique password for each account they have. If a breach of an account occurs, the user can quickly change the password for the one account that has been breached without having to worry that the password was used on other accounts.

Conclusion

Time and time again, the news is full of password breaches. Though it may seem hopeless, there are steps that can be taken to provide better protection for the enterprise environment. Organizations should make sure their passwords are stored according to best practices with the hashes salted. Also, organizations should have tough password requirements that account for sufficient length and complexity. The password age should be set to an appropriate length of time in accordance with the organization's security needs. However, organizational password policies are not enough. The users should be trained on how to create proper passwords and how to use a password manager. Additionally, a password manager should be made available to all users. Organizations should also create a password auditing process using password cracking tools. The password hashes should be dumped and tested against the various password crackers on a monthly schedule. If any passwords are cracked the user should be required to change the password and review the training on creating a strong password. Additionally, a script can be created to run a basic check against a hash at the time of creation to see if the password is in known password dictionaries or is weak password that would be vulnerable to a brute force attack. These steps will help educate and challenge users to create strong password which in return will create a more secure environment for the organization.

References

Bonneau, J. (2012). The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. *2012 IEEE Symposium on Security and Privacy* (pp. 538-552). San Francisco, CA: IEEE.

Chao Shen, T. Y. (2016). User practice in password security: An empirical study of real-life passwords in the wild. *Computers & Security*, 130-141.

Documentation. (n.d.). Retrieved from Rainbow Crack: http://project-rainbowcrack.com/documentation.htm

Hashcat Wiki. (n.d.). Retrieved from Hashcat: https://hashcat.net/wiki/

John the Password Ripper. (2015, 6 4). Retrieved from Openwall: http://www.openwall.com/john/doc/

Lee, K. (2014, 6 25). *How to Create a Secure Password You Can Remember Later: 4 Key Methods*. Retrieved from Open Buffer: https://open.buffer.com/creating-a-secure-password/

Richard Shay, S. K. (2016). Designing Password Policies for Strength and Usability. *ACM Transactions on Information and System Security*, 1-34.

Schneier, B. (2014, 2 25). *Choosing a Secure Password*. Retrieved from Boing Boing: https://boingboing.net/2014/02/25/choosing-a-secure-password.html

Stanek, W. R. (n.d.). *Best Practices for Enforcing Password Policies*. Retrieved from Technet Magazine: https://technet.microsoft.com/en-us/magazine/ff741764.aspx

Takahashi, D. (2009, December 15). *RockYou Hacked, 32 million account passwords potentially exposed*. Retrieved from Venture Beat: http://venturebeat.com/2009/12/15/rockyou-hacked-32-million-account-passwords/

Turner, D. (2012, July 17). *SpiderLabs Blog*. Retrieved from Trustwave: https://www.trustwave.com/Resources/SpiderLabs-Blog/Hashcat-Per-Position-Markov-Chains/

Wawro, A. (2011, 5 3). *How to Build Better Passwords Without Losing Your Mind*. Retrieved from PC World: http://www.pcworld.com/article/227023/how_to_build_a_better_password.html