fhwedel
UNIVERSITY OF APPLIED SCIENCES

Seminar IT-Security

# Password Attacks

*Paul-Louis Pröve*

mail@plpp.de

supervised by

Prof. Dr. Gerd Beuster

gb@fh-wedel.de

Hamburg,

February 20, 2017

## Contents

## 1 Introduction

Password attacks are at the edge of accessing someones secrets. By learning to judge the strength of a password and by understanding how hackers execute attacks, users can make better estimations on how safe they are.

The entropy is widely used to measure how safe a password is, but many sources draw inaccurate conclusions between the entropy of a random password and the strength of a password that was chosen by a person. It is important to understand how these two differ and why realistic password strength is often hard to determine.

Todays hardware gives hackers incredibly powerful machines to launch different types of password attacks. Common password patterns lower possible permutations by such a magnitude that even seemingly safe passwords can be successfully attacked. In combination with frequently used passwords and personal information, hackers can further increase the effectiveness of their attacks.

By explaining common terminologies and analysing different datasets we will look at password attacks from the perspective of users, system administrators and hackers. All three benefit by understanding how the others operate in practice.

# 2 Types of Attacks

## 2.1 Online Attacks

Online password attacks are held against logins of network protocols like POP3, SSH or FTP. The target that is being attacked is a remote computer or server on a different network [1]. The rate at which an attack like this can be processed, depends mostly on the protocol, the connection to the server and the hardware of the server itself. Therefore the attacker has only a very limited influence on the attack rate.

Another challenge with online attacks is the fact that not only a password, but also the associated username is needed. If both of these inputs are unknown the number of possible entries is the result of all possible passwords multiplied by all possible usernames.

A tool that can be used for such an attack is hydra [2]. hydra is a command line tool available for Windows and different unix based operating systems. It's free, open source and still under active development. The command for running hydra will look similar to this:

**“hydra -l admin -P passwords.txt mail.domain.com pop3”**

The lowercase l specifies that the username "admin" is already known. The uppercase P stands for a list of passwords that are to be used for the attack. In the end the address of the server as well as the protocol type is entered. -h will give a full list of hydras syntax.

A common countermeasure against online attacks is a maximum number of logins. Since the verification process is done by the server, it would be possible to block specific IPs after a certain number of unsuccessful tries. A human user is probably not affected by a limit of only one login per second or 100 tries a day. An attacker on the other hand will have a much harder time executing password or DDoS attacks on such a server. Although protocols like SSH can be enhanced for a maximum number of logins, it is not enabled by default [3].

## 2.2 Offline Attacks

Offline attacks are held against password hashes like those produced by MD5, SHA1 or SHA512. Before such an attack can be executed the attacker must first gain a copy of the hashes that he wants to crack [1]. One of the criteria for a secure hash function is the fact that the result cannot be reversed to its input [4]. The attacker must therefore calculate hashes for different passwords and check for identical entries in the data he aquired.

The rate at which an offline attack can be run, depends mostly on the hashtype and the hardware of the computer. NTLM and MD5 will be processed much faster than more complex hash functions like SHA512 or PDF encryption hashes. The usage of GPUs has been gaining a lot popularity over the past years, because they are designed for highly parallelised algorithms. The architecture of GPUs compliments the arithmetic operations that are needed for password attacks [5].

A tool that can be used for such an attack is hashcat. Just like hydra it's open source and available for different platforms. It's still under active development and a command for running hashcat can look like this:

**"hashcat -a 0 -m 0 hashes.txt rockyou.txt"**

-a 0 specifies that the attack type is wordlist and -m 0 tells hashcat that the hashes are MD5 entries. hashcat will then calculate the hashes for the passwords in the rockyou.txt file and cross reference them against the entries in hashes.txt. The full list of hashcats syntax can be viewed with the -h command.

Common countermeasures against offline attacks include Stretching and Salting. Password Stretching describes the process of hashing a password multiple times. The output of the hash function is used as the input of the next iteration. The security increases because more processing time per password is needed [6].

With Password Salting every user gets assigned a random value, which is added to the password before hashing it. This means that two users with the same password will have different password hashes in the database. This way an intruder has to attack every user one-by-one and is not able to create a hash table that works with all users [6].

## 3 Entropy

The strength of a password can be described by the number of guesses attackers need to crack it. The more tries an hacker needs to find the password, the more secure it is [7]. Length and element complexity of the password play into account how large this number will be [8]. This results in two problems when trying to measure the security of a password.

For one this number is highly subjective as every attacker would need a different number of guesses. It's difficult to judge how many tries the average hacker might need. Also, this number can only be stated after an attack was successful. Instead the entropy can be used to objectively predict this event and measure the strength of randomly chosen passwords.

$$H = log_2(N^L) \tag{1}$$

The entropy H of a password can be determined based on its length L and the pool size N each element can be chosen from. Increasing the possible complexity of each element or increasing the length will also increase the entropy of a password [9].

| Length of Password (L) | Entropy |
|---|---|
| 6 | 28 |
| 8 | 38 |
| 10 | 47 |
| 12 | 56 |
| 14 | 66 |

Table 1: Entropy based on password length. N = 26

Because of its exponential growth increasing the length of a password will have a larger effect on the entropy than a more complex pool size [9]. Therefore users who are interested in strengthening their passwords should consider making it longer rather than using additional symbol groups like special characters. Increasing both will still get the best results.

| Character Complexity (N) | Entropy |
|---|---|
| 10 | 27 |
| 26 | 38 |
| 36 | 41 |
| 52 | 46 |
| 95 | 53 |

Table 2: Entropy based on a character complexity. L = 8

Passwords that are made up of actual words are called passphrases. Calculating the entropy of “correcthorsebatterystaple” using the entropy equation based on single characters results in 118. However instead of analysing this passphrase based on characters, one could also do it based on the four words it consists of. “correct”, “horse”, “battery” and “staple” are all in the top 2000 of the most common english words [15]. Instead of seeing a tuple made of 25 lowercase characters, it is also possible to see a tuple with the length of 4 and a pool size of 2000. The same entropy equation is used, but since this is based on a different model the result changes — in this case to 44.

$$H_{chars} = log_2(26^{25}) = 118 \tag{2}$$

$$H_{words} = log_2(2000^4) = 44 \tag{3}$$

The actual entropy of this password depends on the method that was used to generate it. If “correcthorsebatterystaple” is the result of 25 randomly chosen lowercase characters than its entropy would be 118. If instead it was created by using 4 random words from the 2000 most common words than the entropy would be 44. However, since we should predict the strength of a password from the perspective of an attacker, the entropy cannot be higher than the lowest rating we can find to calculate it.

Another problem emerges when taking into account passwords that were chosen by humans. These passwords cannot be seen as truly random, because every person will include some level of self revelation. In theory it is still possible to use the entropy equation to calculate its strength. In practice it is almost impossible to choose N and L in a way that accounts for the subjectivity of the person.

One approach to this gap between the entropy of a random password and the realistic strength is used by a software called zxcvbn [14]. It tries to account for patterns and predictable information users might include in their passwords. For example "password" has an entropy of 38 calculated on a character basis and an entropy of 9 when analysed as a word. However zxcvbn rates its entropy as 0, because its known as the most common password in the world.

## 4 Brute-Force

Systematically enumerating over all possible inputs and testing each as a solution is called a brute-force search. Cracking a password by guessing inputs is tedious work for a human being. If one could do it at a rate of 1 password per second, it would still take 10 days for trying all possible combinations of just 3 characters. Any computer available today can do the same in under a second.

A security company called Sagitta is currently selling a computer for $22.499 that ships with 8 NVidia GTX 1080 included. This machine is called Brutalis and is benchmarked to run at 200 billion MD5 Hashes per second [10]. The time it takes such a machine to attack different lengths of passwords can be summarised as follows.

| Length of Password | Entropy | Runtime |
|---|---|---|
| 6 | 39 | 4 seconds |
| 7 | 46 | 6 minutes |
| 8 | 53 | 9 hours |
| 9 | 59 | 36 days |
| 10 | 66 | 9 years |

Table 3: Sagitta Brutalis (NVidia GTX 1080) cracking all character combination on MD5 hashes with hashcat v3.00

A strong 8 character password consisting of any of the 95 printable ASCII characters will be cracked within 9 hours of using such a machine. Using more complex Hashtypes than MD5 changes these numbers drastically.

| Hashtype | Runtime |
|---|---|
| MD5 | 9 hours |
| SHA1 | 30 hours |
| SHA256 | 4 days |
| 1Password | 9 years |
| Keypass | 1.000 years |
| VeraCrypt (HMAC-SHA512) | 40.000 years |

Table 4: Sagitta Brutalis (NVidia GTX1080) calculating all 8 character combinations with hashcat v3.00

A password protecting a VeraCrypt (HMAC-SHA512) Container can have 25 bits lower entropy than a password hashed with MD5 to offer a comparable level of security, because the hashing algorithm is that much slower.

# 5 Patterns

Identifying password patterns helps hackers to make their attacks more effective. The two most common ones are topologies and keyboard patterns. Other popular patterns are dates, repeats, sequences and l33t speak [14].

## 5.1 Topologies

The topology of a password describes the overall structure of character types that are being used at specific positions. If a pure brute-force attack seems unlikely to be successful, the attacker might only test for passwords that follow a specific set of rules. Characters are most often grouped in one of four categories.

- u: uppercase letter (26)
- l: lowercase letter (26)
- d: numeric digits (10)
- s: special character (33)

The password "Secret01!" uses all four of these categories resulting in an entropy of 59 based on characters. Its topology is "ullllldds" describing the overall structure of its character set. If this topology is known or assumed by an attacker the entropy drops to 40 resulting in roughly 500,000 times less possible passwords.

Since this is the result of just 1 common password pattern, let's have a look at a distribution amongst real user data.

| Topology | Frequency |
|---|---|
| llllllll | 9% |
| llllll | 8% |
| dddddd | 6% |
| lllllll | 6% |
| dddddddd | 4% |
| lllllldd | 3% |

Table 5: Frequency of the top password topologies in the top 1 million worst passwords [11]

The top 11 topologies make up 50% of all passwords. Since this is a collection of the 1 million worst passwords, here's a different dataset containing 14.3 million passwords.

| Topology | Frequency |
|---|---|
| lllllll | 4% |
| llllll | 4% |
| lllllll | 4% |
| lllllllll | 3% |
| ddddddd | 3% |
| dddddddddd | 3% |

Table 6: Frequency of the top password topologies in the RockYou password list [12]

The frequency of the top three most common topologies has dropped by 50%, but the overall order appears to be similar. In fact, 16 out of the top 20 patterns of each are completely the same. 18 out of 20 can be found by using only lowercase letters and numbers. The top 22 topologies make up 50% of all passwords.

One approach to increase security is telling users to include lowercase letters, uppercase letters and numbers in their passwords. Another dataset gives an insight of this theory.

| Topology | Frequency |
|---|---|
| ullllldd | 13% |
| ulllllldd | 13% |
| ullldddd | 11% |
| ullllllldd | 7% |
| ulllldddd | 5% |

Table 7: Frequency of the most common password topologies in anonymous corporate data by KoreLogic [13]

These top 5 topologies result in 50% of all user passwords. While the average entropy of each password increased, users got less creative with their choice of topologies. The patterns that describe the top 50% have a combined entropy of 45 in comparison to an entropy of 57 it would take to crack half of the RockYou passwords.

## 5.2 Keyboard Patterns

Keyboard patterns describe easily repeatable "walks" from one key to the next on a keyboard. The 29th most common password "1qaz2wsx" [11] seems much too random to be even in the Top 100. But when it's typed using a standard QWERTY-Keyboard the pattern becomes obvious.

Keyboard patterns are frequently used throughout the most common passwords. However, in reality they don't pose a big thread [16]. Tools like zxcvbn, which itself was named after a keyboard pattern, can easily recognise these weak passwords and stop people from using them.

# 6 Password Lists

Lists of passwords are mainly used for two different purposes. They can be analysed for length, patterns or other criteria, like we have done in the previous chapter. The second purpose is inputting them into cracking tools to run a brute-force attack exclusively on the included words. This is called a dictionary attack.

Password lists can consist of leaked passwords, common names, actual dictionaries or a mixture of the three. Popular lists among hackers are:

- RockYou-List
- 10 Million Passwords by Mark Burnett
- Leaked user passwords
- English dictionary
- Facebook first names

Most of them don't exceed 100 million passwords and can often be tested within a few seconds. That's why many attackers combine these entries to chains of passwords also called passphrases.

The entire english dictionary includes roughly 170,000 different words [18]. If we base a brute-force attack on this dictionary, every combination of two random words has an entropy of 35. Increasing the chain to three words results in 52 and four words represent an entropy of 70. Many times however users choose passphrases based on commonly used words. A six word passphrase built around the 100 most common words has an entropy of 40.

# 7 Personalisation

Personal information make it into the passwords of many users. 136 out of the 250 most common passwords include names, cities, sport teams, brands or activities [11]. Without knowing the people these passwords belong to, it is still safe to assume that these words have some kind of meaning to these users.

In the time of social networks and sharing all kinds of media online, it is often quite easy to collect information people might use to create their passwords. A person of interest who is an intensive Facebook user, gives potential attackers easy access to information about relatives, friends, birthdays, work, nicknames or interests. With a little more effort it is possible to create a web crawler that analyses posts and comments of that person and detects commonly used words.

CUPP is an open source command line tool written in Python [17]. It allows attackers to enter different kinds of information about a person like names, dates, pets or family members. Afterwards it will generate a personalised password list based on these information. CUPP has many options to tweak the result. For example there's "1337 Mode" that will switch certain letters like "E" for numbers that look similar like "3".

Personalised password attacks sit on the other side of the spectrum when comparing them to high volume database attacks. They increase they likeliness of cracking a single users password at the cost of time spent on information gathering.

# 8 Summary

Security experts and hackers have spent a lot of time investigating how average people create their passwords. They are one step ahead of those who have only little knowledge of information theory and statistics. Passwords that seem strong and random to an user can often be easily cracked with todays computer hardware.

The entropy is good indicator for the strength of a password, given that it was randomly generated. Humans however are not very good at making truly random decisions. Their passwords are often influenced by the world around them. This is where the entropy gets inaccurate. Tools like zxcvbn fill the gap of making realistic security assumptions about personalised passwords.

From the perspective of an user one should try to use random password generators from trustworthy sources and have a different password for each use case. In combination with password managers that keep track of accounts, one can stay assumably safe without complicating their lives. For passwords that the user makes up on their own, it is best to validate the password strength by a tool like zxcvbn. There are a lot of other password validators that do not understand the difference between the entropy and the realistic strength of a personalised password.

From the perspective of a security officer new problems can evolve when forcing people to use a specific set of character types. Instead of judging a password based on its character complexity, its much more useful to rate the strength of the password as a whole. Other password related measures can be made outside the actual choice of the passwords. Stretching and Salting secure against the usage of hash tables and decrease possible attack rates.

From the perspective of an hacker one should differentiate between attacking a large set of passwords and attacking a single user. Even large password lists can be processed rather quickly with computers available today. Afterwards the attack can be expanded to pure brute-force and dictionary attacks until the effort stops being feasible. These attacks can be made more efficient by using common topologies and patterns. Attacking a single user puts a lot more work into the gathering of information than the execution of the attack itself.

In the end a password needs to be more secure than the value of information it is protecting, so that even the most skilled hacker has no feasible reason to continue the attack.

## References

[1] Georgia Weidman. *Penetration Testing: A Hands-On Introduction to Hacking.* No Starch Press, 1st Edition, 2014

[2] THC Hydra - Tool for Online Password Attacks `https://github.com/vanhauser-thc/thc-hydra`. Accessed 20 Feb. 2017.

[3] Manpagez - SSH daemon configuration file `http://www.manpagez.com/man/5/sshd_config/`. Accessed 20 Feb. 2017.

[4] M. Naor. *Universal one-way hash functions and their cryptographic applications.* ACM New York, NY, USA, 1989

[5] Martijn Sprengers. *Speeding up GPU-based password cracking.* `http://2012.sharcs.org/slides/sprengers.pdf`. Accessed 20 Feb. 2017.

[6] Jeremiah Blocki, Anirudh Sridhar. *Client-CASH: Protecting Master Passwords against Offline Attacks.* ASIA CCS 2016, 10.1145/2897845.2897876

[7] Wanli Ma, John Campbell, Dat Tran, Dale Kleeman. *Password Entropy and Password Quality.* 2010 Fourth International Conference on Network and System Security, DOI 10.1109/NSS.2010.18.

[8] Cyber Security Tip ST04-002. *Choosing and Protecting Passwords.* US CERT. Retrieved June 20, 2009

[9] Johannes Weber. *Password Strength/Entropy: Characters vs. Words.* `https://blog.webernetz.net/2013/07/30/password-strengthentropy-characters-vs-words/`. Accessed 20 Feb. 2017.

[10] Sagitta Brutalis. *8x NVidia GTX 1080 Hashcat Benchmarks.* `https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40`. Accessed 20 Feb. 2017.

[11] Daniel Miessler. *SecList: 10 Million Password List - Top 1 Million.* `https://github.com/danielmiessler/SecLists/blob/master/Passwords/10_million_password_list_top_1000000.txt`. Accessed 20 Feb. 2017.

[12] Daniel Miessler. *SecList: RockYou List.* `https://github.com/danielmiessler/SecLists/blob/master/Passwords/rockyou.txt.tar.gz`. Accessed 20 Feb. 2017.

[13] Hank Leininger, KoreLogic. *Password Topology Histogram Wear-Leveling.*
https://www.korelogic.com/Resources/Presentations/bsidesavl_pathwell_2014-06.pdf. Accessed 20 Feb. 2017.

[14] Dan Wheeler, Dropbox. *Zxcvbn - A realistic password strength estimator.*
https://github.com/dropbox/zxcvbn. Accessed 20 Feb. 2017.

[15] Josh Kaufman. *Google 10000 most common english words.*
https://github.com/first20hours/google-10000-english. Accessed 20 Feb. 2017.

[16] wpengine. *Unmasked: What 10 million passwords reveal about the people who choose them.*
http://wpengine.com/unmasked/. Accessed 20 Feb. 2017.

[17] Mebus. *CUPP: Common User Passwords Profiler.*
https://github.com/Mebus/cupp. Accessed 20 Feb. 2017.

[18] Oxford Dictionaries *How many words are there in the English language?*
https://en.oxforddictionaries.com/explore/how-many-words-are-there-in-the-english-language. Accessed 20 Feb. 2017.