Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies in Bratislava

Principles of information security

Freely distributed password cracking tools

Ákos Lévárdy

March 17, 2024

Contents

1 Topic of the project: Freely distributed password cracking tools

In my topic I will be focused on freely distributed password cracking tools. Firstly, I will describe how passwords are encrypted or hashed. Passwords are used everywhere for user authentication, which is a widely adopted method due to its intuitive logic and ease of implementation for developers.

Despite their popularity, passwords pose security risks as password crackers are crafted to extract credentials from compromised data obtained through breaches or hacks.

2 What is password cracking?

Most password-based authentication systems do not store a user's actual password. Instead, they store a password hash, which is the result of sending the password and a random value called a salt through a hash function.

I will compare different hash functions and describe how they work when in use. Basically, hash functions are designed to be one way. This means that it is very difficult to get the original password from the hashed output.

It is important to know that hash functions are producing the same output for the same input. Comparing two hashes of a password is eventually the same as comparing two real passwords.

2.1 Password cracking tools

There are many different tools for password cracking.
The most popular ones are:

- Hashcat
- John the Ripper
- Brutus
- Wfuzz
- THC Hydra
- Meduza
- RainbowCrack
- OphCrack
- L0phtCrack
- Aircrack-ng

I will be comparing two of these password cracking tools – Hashcat and John the Ripper, analyze them, see how they work and then compare their performance.

3 Types of attacks on passwords

There are various types of attacks on passwords, and they can be categorized into different methods based on their techniques.
Most common types of password attacks:

- Brute Force Attack - This involves trying all possible combinations of passwords until the correct one is found.

- Dictionary Attack - Attackers use a predefined list of common words, dictionaries to try and guess the password.
- Rainbow Table Attack - A precomputed table containing the hash values of commonly used passwords. This attack compares these hashes with the target system's stored password hashes to find a match.
- Phishing – This attack tries to trick individuals into revealing their passwords by posing as a trustworthy entity.
- Keylogging - Malicious software or hardware records keystrokes without the user noticing it and capturing sensitive information such as usernames and passwords.
- Man-in-the-Middle (MitM) Attack - Intercepting communication between two sides to capture information, including passwords and usernames.
- Shoulder Surfing - Observing someone entering their password without their knowledge.

4 Progress report 1

For the first progress report I will analyse the different hashing algorithms and compare the types of attacks on passwords. I will write about the history of some tools for password cracking and about how to defend our passwords with password managers and other tools.

5 Progress report 2

For the second progress report I will test two password cracking tools and then compare how they work and their performance. I will use John the Ripper and Hashcat on Kali Linux after I set up a Virtual box for it.

6 Authentication

For authenticating a user there are 2 different types. One-factor authentication relies solely on one type of information for verification (usually a password), two-factor authentication enhances security by requiring two different types of information, making it harder for unauthorized users to gain access even if they have obtained one factor (password).

- One-Factor Authentication (1FA):
  - In one-factor authentication, only one type of information is required to verify the identity of the user.
  - Typically, this involves something the user knows, such as a password or a PIN.
  - It's the simplest form of authentication and is commonly used in many basic systems.
- Two-Factor Authentication (2FA):
  - Two-factor authentication requires two different types of information to authenticate the user.
  - These factors usually fall into one of three categories: something the user knows (password), something the user has (smartphone or a security token), or something the user is (biometric data like fingerprints or facial recognition).
  - By requiring two factors, 2FA adds an extra layer of security beyond just a password, making it more difficult for unauthorized users to gain access.
  - Common implementations of 2FA include receiving a text message with a code, using an authenticator app, or using biometric authentication along with a password.

7 Password-Hashing Schemes - PHS

Everyday tasks involve the use of computers, and many people use the services that are offered. The most popular method for authenticating users on the web is one-factor authentication, which consists of a password and a username.

Unfortunately, attackers take advantage of weak password management procedures to reveal users' credentials, which hurts both users and providers. In the majority of these cases, the user data was either processed directly by a cryptographic hash function or kept in cleartext.

7.1 Password-Hashing procedures

We use password-hashing procedures to secure this user-related data. Currently, the Password-Based Key Derivation Function 2 (PBKDF2) standard primitive is

in use, however other popular schemes like Blowfish cipher - Bcrypt and Scrypt are also in use. The development of parallel computing has made it possible for multiple password-hash cracking attempts.

For the purpose of developing new, widely-acceptable password-hashing algorithms that are more safe and efficient, the global cryptography community organized the Password Hashing Competition (PHC). PHC improved our understanding of hashing passwords. Security flaws were discovered through more investigation, and new methods were subsequently developed.

7.2 Background Theory

A password is a secret that is straightforward for the user to remember and consists of just a few printed characters. In computer systems, passwords are frequently used for user authentication. For each account, the system keeps track of the user's identity and password. The individual then uses this information to log in and access the service.

The creation of cryptographic keys is another use for passwords. Based on an input password, the Key-Derivation Functions (KDF) generates a variety of cryptographic keys. KDFs primarily serve for session data encryption, and cryptographic keys are applied by encryption/decryption functions.

Cryptographic keys play a critical role in ensuring the security of data and communications in cryptographic systems. They enable encryption, decryption, digital signing, and authentication, helping to protect sensitive information from unauthorized access, tampering, or interception.
Proper key management practices are essential to maintaining the effectiveness and security of cryptographic systems.

Encryption converts plaintext into ciphertext using an encryption algorithm and a key, while decryption reverses this process to recover the original plaintext. These processes play a crucial role in protecting data confidentiality and integrity, ensuring that only authorized users can access and understand the encrypted information.

7.3 Key-Derivation Function

A key derivation function (KDF) is a cryptographic tool used to derive one or more secret keys from a single, fixed-length secret value, such as a master key or a passphrase.

- The KDF takes as input the secret value and possibly some additional parameters, such as a salt (random value) or context information.
- The KDF expands the input secret into one or more longer keys. This expansion process typically involves applying a cryptographic hash function or a more complex cryptographic algorithm repeatedly to the input, potentially mixing in the salt or other parameters. The goal is to generate keys that are unpredictable and have high entropy (randomness).

- The output of the KDF is one or more derived keys, which can be used for various cryptographic purposes, such as encryption, authentication, or generating other cryptographic parameters.
- Security Properties:
  A strong KDF should include the following security features:
  - Key Derivation: Both the derived keys and the input secret must be computationally independent of one another.
  - Resistance to Attacks: The KDF should resist various cryptographic attacks, such as brute-force attacks (trying all possible inputs) and pre-image attacks (deriving the input from the output).
  - Salt and Context Sensitivity: Incorporating a salt and context information can enhance the security of the derived keys by making them unique to each application or usage scenario.
- Essentially, a key derivation function offers a safe method for producing cryptographic keys from a single secret value, guaranteeing the confidentiality and integrity of the derived keys and making them appropriate for usage in cryptographic protocols and applications.

7.4 Password Security Risks and Vulnerabilities

When people create passwords for their accounts, they often use simple ones that are just eight characters long. These passwords are not very secure because they're easy to guess.

Hackers can use a method called exhaustive search to try every possible combination of characters until they find the right password. Once they do, they can access the account just like the real user. Even graphical passwords, where you draw patterns or pick images, can be easy for hackers to figure out because they don't provide much security.

7.5 Password-Based Key Derivation Function 2 - PBKDF2

Hashing-based techniques are the standard option for establishing password protection. The primary methods for making KDFs possible are keyed-hash message authentication codes (HMACs) and cryptographic hash functions that translate a password into one or more secret keys.

PBKDF2 uses pseudo-random functions (PRF), commonly implemented by HMACs. The SHA-256 function is the standard internal hash option for the HMAC.

The scheme parses the password and the salt as inputs. Attacks utilizing precomputed data, such as dictionary attacks (which test hundreds of plausible combinations to determine the passphrase) and rainbow-table attacks (which make use of tables with precomputed hashes), are made more resilient by the salt. It typically has 8 bytes in size.

PBKDF2 steps:

- The salt and the password are initially processed by the HMAC.
- The derived key is then generated by processing the data multiple times.
- Although 1000 iterations is the minimum required amount, given that the guideline was set in 2000, it is no longer deemed sufficient.
- The password and the previous HMAC result are parsed by the HMAC at each repetition.
- The HMAC outcomes are then XORed.
- The outcome of the XOR operation in the last round is the final hash.

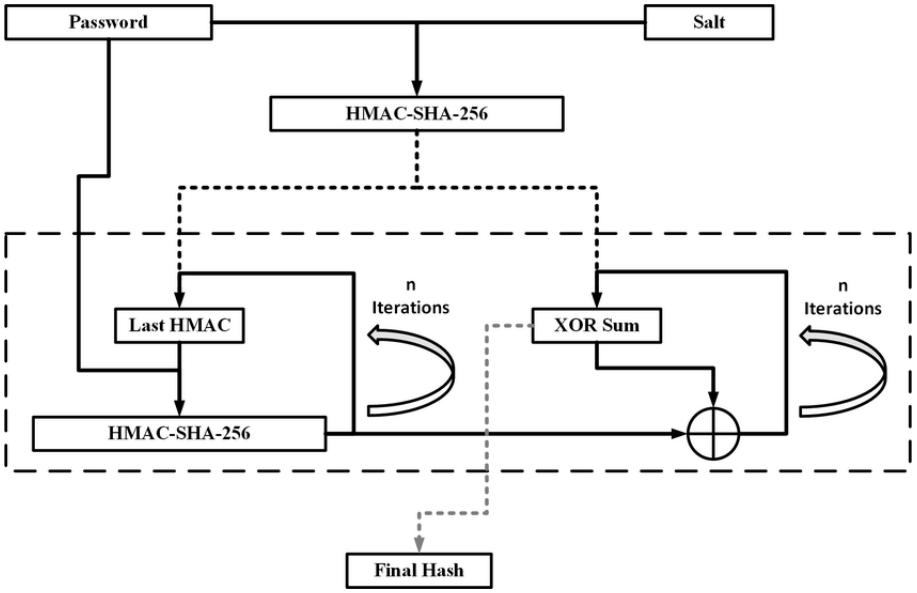

Figure 1: PBKDF2

The effective hardware-based cracking has a downside because it can be created as a small hardware implementation with little RAM requirements. Therefore, on GPUs, FPGAs, and ASICs, low-cost brute-force attacks are possible. In roughly one week, the most rapid attacks can crack about 65% of common passphrases. Approximately 245,000 passwords are obtained per second by attacks on multi-FPGA devices.

7.6 Bcrypt

Bcrypt is the chosen password hashing system for the BSD operating system. It makes use of the Blowfish block cipher. By default, it processes passwords that are 56 bytes long and produces hashes that are 24 bytes long. The number of iterations increases exponentially as the computational power of potential attackers grows, ensuring adequate protection against brute-force attacks. Additionally, a 16-byte salt value is employed to strengthen defenses against attacks relying on rainbow tables.
Bcrypt steps:

- Initialization: Bcrypt starts by setting up the Blowfish cipher with a cost parameter, a salt (a random value), and your password. This sets up a complex system of encryption.
- Key Schedule: Bcrypt then creates a set of subkeys based on your password. These subkeys are mixed with the original password in a special way.
- Encryption: The salt is encrypted using the key schedule. This process repeats several times, making it harder for attackers to reverse-engineer the password.
- Magic Value Encryption: A special value is encrypted multiple times using Blowfish. This further scrambles the data.
- Final Hash: The final hash is created by combining the salt, cost, and the result of the encryption process

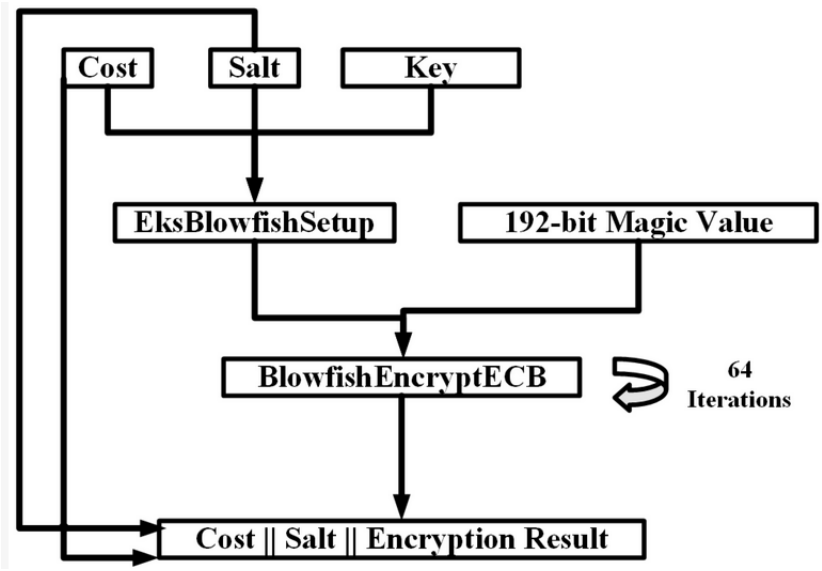

Figure 2: Bcrypt

The method uses 4KB of RAM and outperforms PBKDF2 in terms of blocking parallel cracking. While the best attack discovers 20,583 passwords per second, the least expensive attack only manages 1207 cracks per second at a cost of $99.

7.7 Scrypt

Scrypt is designed to be a memory-hard password hashing scheme, which means it makes it difficult for attackers to use large amounts of memory when trying to crack passwords.
Scrypt steps:

- It is based on the PBKDF2 algorithm and the Salsa stream cipher.
- It has features like MFcrypt (Memory-Hard Function), ROMix (Rounds Mixer), BlockMix, and SMix (Sequential Memory-Hard Function).
- To make sure all of the random values are kept in memory, ROMix creates a bunch of them and shuffles them around.
- BlockMix facilitates this procedure by utilizing another version of the Salsa cipher.
- SMix uses one or more ROMix functions to further complicate matters.
- The number of ROMix functions (parallelization parameters) determines how much time and space the process takes.
- MFcrypt is the final step, where the key-derivation function combines the mixing function with PBKDF2 using the SHA-256 hash function.

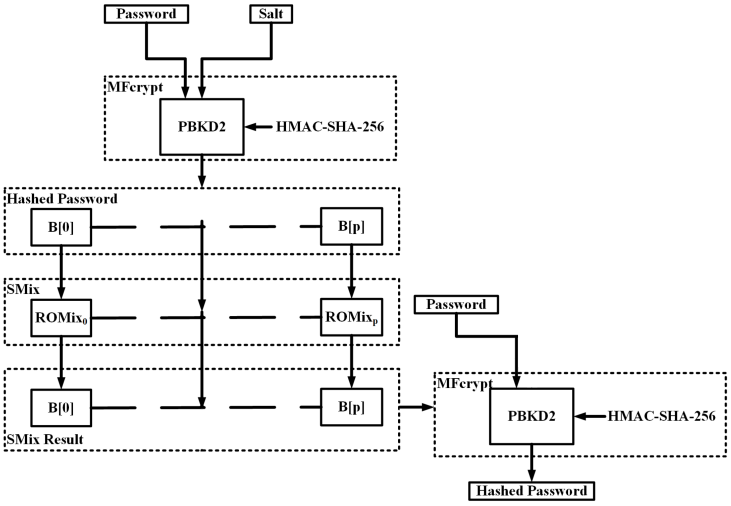

Figure 3: Scrypt

8 Password cracking tools

Passwords are created by humans using recurring patterns. These patterns are taken advantage of by data-driven password cracking techniques that rely on massive password breach collections.

Nowadays, it is common practice to simulate password cracking in order to determine the strength of a password. Numerous algorithms for breaking passwords are probabilistic, which means they build a model and then give each potential password a probability. The strength of a password is proportional to the number of passwords with higher probability based to that model, assuming the attacker logically guesses passwords in descending order of likelihood.

However, probabilistic tools are rarely used by real-world attackers; instead, they mostly employ programs like Hashcat and John the Ripper (JtR).

They give practical explanations. In offline attacks, the time it takes to make and check a guess includes the time to create a guess, hash it, and see if it matches the target password. Probabilistic algorithms are good at guessing passwords, but they take a lot of computer power to generate each guess.

So, for most fast hash functions, it's quicker to crack passwords using different software tools. While the likelihood of guessing a password with a probabilistic model might match the order of guessing with software tools, it's not always accurate.

The mangled-wordlist assaults of JtR and Hashcat are the most frequently used and intellectually engaging. These attacks take advantage of the fact that passwords typically vary in subtle, predictable ways; for example, a word may have a digit added to it by one person, while the same word may have a symbol added by another. In a mangled-wordlist attack, the attacker generates a rule list of mangling rules (such as appending a digit and replacing "s" with "$") written in a transformation language given by the tool, together with a wordlist comprising popular passwords and natural language content. Every mangling rule is applied to every word in the order that the input lists specify during the whole attack.

8.1 HASHCAT

Hashcat is a popular password cracking tool used by security professionals and attackers to recover lost or forgotten passwords or to test the strength of passwords.

Types of attacks:

- Brute-Force Attack: Hashcat can perform a brute-force attack, where it systematically tries every possible combination of characters until it finds the correct password. This method is effective but can be time-consuming, especially for longer and more complex passwords.
- Dictionary Attack: Instead of trying every possible combination, Hashcat can use a predefined list of commonly used passwords, called a dictionary, to attempt to crack passwords. This method is faster than brute-force but relies on the likelihood that the password is contained in the dictionary.
- Combination Attack: Hashcat can also combine both brute-force and dictionary attacks. It starts with the words in the dictionary and then appends, prepends, or replaces characters to create variations of those words, effectively expanding the search space.
- Rule-Based Attack: Hashcat supports rule-based attacks where users can define custom rules to manipulate the dictionary words before attempting them as passwords. For example, rules can be created to add numbers or special characters to the end of dictionary words.
- Mask Attack: This method allows users to specify a mask or template for the password, defining the possible characters and their positions. Hashcat then generates and tests passwords based on this mask.
- Hybrid Attack: Hashcat can combine different attack modes to increase the chances of cracking passwords. For example, it can perform a dictionary attack with rule-based mutations.

8.2 John the Ripper - JTR

John the Ripper is another widely used password cracking tool utilized by security professionals and attackers alike. It operates using similar techniques as Hashcat, employing various methods to crack passwords. It is a versatile and powerful tool for cracking passwords, offering various techniques to suit different scenarios and types of passwords.

Types of attacks:

- Dictionary Attack: John the Ripper can perform a dictionary attack by trying words from a predefined list of common passwords or from a custom dictionary file. It systematically checks each word in the dictionary against the hashed passwords to see if there's a match.
- Brute-Force Attack: Like Hashcat, John the Ripper can also conduct brute-force attacks by systematically trying every possible combination of characters until it finds the correct password. This method is effective but can be time-consuming, especially for longer and more complex passwords.
- Rule-Based Attack: John the Ripper supports rule-based attacks where users can define custom rules to manipulate the dictionary words before attempting them as passwords. For example, rules can be created to add numbers or special characters to the end of dictionary words.
- Incremental Mode: This mode in John the Ripper generates passwords by incrementally increasing their length and complexity based on predefined rules. It starts with short passwords and gradually builds up to longer ones, trying different combinations along the way.
- Hybrid Attack: John the Ripper can combine different attack modes, such as dictionary attacks with rule-based mutations or brute-force attacks with specific character sets.
- Single Crack Mode: This mode allows John the Ripper to crack password hashes with just one attempt, using various techniques like dictionary, brute-force, and incremental attacks

9 Password Managers

10 Strong Passwords

References

[1] George Hatzivasilis. Password-hashing status. *Cryptography*, 1(2), 2017. doi:10.3390/cryptography1020010.

[2] Enze Liu, Amanda Nakanishi, Maximilian Golla, David Cash, and Blase Ur. Reasoning analytically about password-cracking software. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 380–397, 2019. doi:10.1109/SP.2019.00070.

[3] Hamza Touil, Nabil El Akkad, Khalid Satori, Naglaa F. Soliman, and Walid El-Shafai. Efficient braille transformation for secure password hashing. *IEEE Access*, 12:5212–5221, 2024. doi:10.1109/ACCESS.2024.3349487.