Password cracking

By Contributors to Wikimedia projects

Published: 2004-02-20 · Archived: 2026-04-05 15:03:03 UTC

From Wikipedia, the free encyclopedia

In cryptanalysis and computer security, **password cracking** is the process of guessing passwords[1] protecting a computer system. A common approach (brute-force attack) is to repeatedly try guesses for the password and to check them against an available cryptographic hash of the password.[2] Another type of approach is **password spraying**, which is often automated and occurs slowly over time in order to remain undetected, using a list of common passwords.[3]

The purpose of password cracking might be to help a user recover a forgotten password (due to the fact that installing an entirely new password would involve System Administration privileges), to gain unauthorized access to a system, or to act as a preventive measure whereby system administrators check for easily crackable passwords. On a file-by-file basis, password cracking is utilized to gain access to digital evidence to which a judge has allowed access, when a particular file's permissions restricted.

Brute-force password guessing attacks against online systems are usually ineffective because systems are designed to lock accounts after a certain number of unsuccessful login attempts. Obtaining the cryptographic hash of the password makes it possible to perform an unlimited number of offline guesses. Hash files can might be obtained by third parties by court-ordered access, or as part of an illegal data breach. If passwords are reused across web sites, the hash file obtained from one web site can be used for cracking, and then the discovered passwords can be attempted against the same email addresses on other web sites.

The password spraying approach evades account lockout countermeasures by guessing the same password across many accounts, allowing time to pass between attempts on the same account, which might include legitimate logins. This approach does not target specific accounts.

Time needed for password searches

[edit]

The time to crack a password is related to bit strength, which is a measure of the password's entropy, and the details of how the password is stored. Most methods of password cracking require the computer to produce many candidate passwords, each of which is checked. One example is brute-force cracking, in which a computer tries *every* possible key or password until it succeeds. With multiple processors, this time can be optimized through searching from the last possible group of symbols and the beginning at the same time, with other processors being placed to search through a designated selection of possible passwords.[4] More common methods of password cracking, such as dictionary attacks, pattern checking, and variations of common words, aim to optimize the number of guesses and are usually attempted before brute-force attacks. Higher password bit strength

exponentially increases the number of candidate passwords that must be checked, on average, to recover the password and reduces the likelihood that the password will be found in any cracking dictionary.[5]

The ability to crack passwords using computer programs is also a function of the number of possible passwords per second which can be checked. If a hash of the target password is available to the attacker, this number can be in the billions or trillions per second, since an *offline attack* is possible. If not, the rate depends on whether the authentication software limits how often a password can be tried, either by time delays, CAPTCHAs, or forced lockouts after some number of failed attempts. Another situation where quick guessing is possible is when the password is used to form a cryptographic key. In such cases, an attacker can quickly check to see if a guessed password successfully decodes encrypted data.

For some kinds of password hash, ordinary desktop computers can test over a hundred million passwords per second using password cracking tools running on a general purpose CPU and billions of passwords per second using GPU-based password cracking tools[1][6][7] (see John the Ripper benchmarks).[8] The rate of password guessing depends heavily on the cryptographic function used by the system to generate password hashes. A suitable password hashing function, such as bcrypt, is many orders of magnitude better than a naive function like simple MD5 or SHA. A user-selected eight-character password with numbers, mixed case, and symbols, with commonly selected passwords and other dictionary matches filtered out, reaches an estimated 30-bit strength, according to NIST. 2^{30} is only one billion permutations[9] and would be cracked in seconds if the hashing function were naive. When ordinary desktop computers are combined in a cracking effort, as can be done with botnets, the capabilities of password cracking are considerably extended. In 2002, distributed.net successfully found a 64-bit RC5 key in four years, in an effort which included over 300,000 different computers at various times, and which generated an average of over 12 billion keys per second.[10]

Graphics processing units can speed up password cracking by a factor of 50 to 100 over general purpose computers for specific hashing algorithms. As an example, in 2011, available commercial products claimed the ability to test up to 2,800,000,000 NTLM passwords a second on a standard desktop computer using a high-end graphics processor.[11] Such a device can crack a 10-letter single-case password in one day. The work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs. However some algorithms run slowly, or even are specifically designed to run slowly, on GPUs. Examples are DES, Triple DES, bcrypt, scrypt, and Argon2.

Hardware acceleration in a GPU has enabled resources to be used to increase the efficiency and speed of a brute force attack for most hashing algorithms. In 2012, Stricture Consulting Group unveiled a 25-GPU cluster that achieved a brute force attack speed of 350 billion guesses of NTLM passwords per second, allowing them to check 95^8 password combinations in 5.5 hours, enough to crack all 8-character alpha-numeric-special-character passwords commonly used in enterprise settings[*citation needed*]. Using ocl-Hashcat Plus on a Virtual OpenCL cluster platform,[12] the Linux-based GPU cluster was used to "crack 90 percent of the 6.5 million password hashes belonging to users of LinkedIn".[13]

For some specific hashing algorithms, CPUs and GPUs are not a good match. Purpose-made hardware is required to run at high speeds. Custom hardware can be made using FPGA or ASIC technology. Development for both technologies is complex and (very) expensive. In general, FPGAs are favorable in small quantities, ASICs are

favorable in (very) large quantities, more energy efficient, and faster. In 1998, the Electronic Frontier Foundation (EFF) built a dedicated password cracker using ASICs. Their machine, Deep Crack, broke a DES 56-bit key in 56 hours, testing over 90 billion keys per second.[14] In 2017, leaked documents showed that ASICs were used for a military project that had a potential to code-break many parts of the Internet communications with weaker encryption.[15] Since 2019, John the Ripper supports password cracking for a limited number of hashing algorithms using FPGAs.[16] Commercial companies are now using FPGA-based setups for password cracking.[17]

Easy to remember, hard to guess

[edit]

Passwords that are difficult to remember will reduce the security of a system because:

- users might need to write down or electronically store the password using an insecure method,
- users will need frequent password resets, and
- users are more likely to re-use the same password.

Similarly, the more stringent the requirements for password strength, e.g. "have a mix of uppercase and lowercase letters and digits" or "change it monthly", the greater the degree to which users will subvert the system.[18]

In "The Memorability and Security of Passwords",[19] Jeff Yan *et al.* examine the effect of advice given to users about a good choice of password. They found that passwords based on thinking of a phrase and taking the first letter of each word are just as memorable as naively selected passwords, and just as hard to crack as randomly generated passwords. Combining two unrelated words is another good method. Having a personally designed "algorithm" for generating obscure passwords is another good method.

However, asking users to remember a password consisting of a "mix of uppercase and lowercase characters" is similar to asking them to remember a sequence of bits: hard to remember, and only a little bit harder to crack (e.g. only 128 times harder to crack for 7-letter passwords, less if the user simply capitalizes one of the letters). Asking users to use "both letters and digits" will often lead to easy-to-guess substitutions such as 'E' → '3' and 'I' → '1': substitutions which are well known to attackers. Similarly, typing the password one keyboard row higher is a common trick known to attackers.

Research detailed in an April 2015 paper by several professors at Carnegie Mellon University shows that people's choices of password structure often follow several known patterns. For example, when password requirements require a long minimum length such as 16 characters, people tend to repeat characters or even entire words within their passwords.[20] As a result, passwords may be much more easily cracked than their mathematical probabilities would otherwise indicate. Passwords containing one digit, for example, disproportionately include it at the end of the password.[20]

On July 16, 1998, CERT reported an incident where an attacker had found 186,126 encrypted passwords. By the time the breach was discovered, 47,642 passwords had already been cracked.[21]

In December 2009, a major password breach of Rockyou.com occurred that led to the release of 32 million passwords. The attacker then leaked the full list of the 32 million passwords (with no other identifiable information) to the internet. Passwords were stored in cleartext in the database and were extracted through an SQL injection vulnerability. The Imperva Application Defense Center (ADC) did an analysis on the strength of the passwords.[22] Some of the key findings were:

- about 30% of users chose passwords whose length was below seven characters,
- almost 60% of users chose their passwords from a limited set of alpha-numeric characters, and
- nearly 50% of users used names, slang words, dictionary words, or trivial passwords that employed weak constructs such as consecutive digits and/or adjacent keyboard keys—case in point, the most common password among RockYou account owners was simply “123456”.[22]

In June 2011, NATO (North Atlantic Treaty Organization) suffered a security breach that led to the public release of first and last names, usernames, and passwords of more than 11,000 registered users of their e-bookshop. The data were leaked as part of Operation AntiSec, a movement that includes Anonymous, LulzSec, and other hacking groups and individuals.[23]

On July 11, 2011, Booz Allen Hamilton, a large American consulting firm that does a substantial amount of work for the Pentagon, had its servers hacked by Anonymous and leaked the same day. "The leak, dubbed 'Military Meltdown Monday', includes 90,000 logins of military personnel—including personnel from USCENTCOM, SOCOM, the Marine Corps, various Air Force facilities, Homeland Security, State Department staff, and what looks like private-sector contractors."[24] These leaked passwords were found to be hashed with unsalted SHA-1, and were later analyzed by the ADC team at Imperva, revealing that even some military personnel used passwords as weak as "1234".[25]

On July 18, 2011, Microsoft Hotmail banned the password: "123456".[26]

In July 2015, a group calling itself "The Impact Team" stole the user data of Ashley Madison.[27] Many passwords were hashed using both the relatively strong bcrypt algorithm and the weaker MD5 hash. Attacking the latter algorithm allowed some 11 million plaintext passwords to be recovered by password cracking group CynoSure Prime.[28]

One method of preventing a password from being cracked is to ensure that attackers cannot get access even to the hashed password. For example, on the Unix operating system, hashed passwords were originally stored in a publicly accessible file `/etc/passwd`. On modern Unix (and similar) systems, on the other hand, they are stored in the shadow password file `/etc/shadow`, which is accessible only to programs running with enhanced privileges (i.e., "system" privileges). This makes it harder for a malicious user to obtain the hashed passwords in the first instance, however many collections of password hashes have been stolen despite such protection. And some common network protocols transmit passwords in cleartext or use weak challenge/response schemes.[29][30]

The use of salt, a random value unique to each password that is incorporated in the hashing, prevents multiple hashes from being attacked simultaneously and also prevents the creation of pre-computed dictionaries such as rainbow tables.

Another approach is to combine a site-specific secret key with the password hash, which prevents plaintext password recovery even if the hashed values are purloined. However privilege escalation attacks that can steal protected hash files may also expose the site secret. A third approach is to use key derivation functions that reduce the rate at which passwords can be guessed.[31]:5.1.1.2

Modern Unix Systems have replaced the traditional DES-based password hashing function crypt() with stronger methods such as crypt-SHA, bcrypt, and scrypt.[32] Other systems have also begun to adopt these methods. For instance, the Cisco IOS originally used a reversible Vigenère cipher to encrypt passwords, but now uses md5-crypt with a 24-bit salt when the "enable secret" command is used.[33] These newer methods use large salt values which prevent attackers from efficiently mounting offline attacks against multiple user accounts simultaneously. The algorithms are also much slower to execute which drastically increases the time required to mount a successful offline attack.[34]

Many hashes used for storing passwords, such as MD5 and the SHA family, are designed for fast computation with low memory requirements and efficient implementation in hardware. Multiple instances of these algorithms can be run in parallel on graphics processing units (GPUs), speeding cracking. As a result, fast hashes are ineffective in preventing password cracking, even with salt. Some key stretching algorithms, such as PBKDF2 and crypt-SHA iteratively calculate password hashes and can significantly reduce the rate at which passwords can be tested, if the iteration count is high enough. Other algorithms, such as scrypt are memory-hard, meaning they require relatively large amounts of memory in addition to time-consuming computation and are thus more difficult to crack using GPUs and custom integrated circuits.

In 2013 a long-term Password Hashing Competition was announced to choose a new, standard algorithm for password hashing,[35] with Argon2 chosen as the winner in 2015. Another algorithm, Balloon, is recommended by NIST.[36] Both algorithms are memory-hard.

Solutions like a security token give a formal proof answer[*clarification needed*] by constantly shifting password. Those solutions abruptly reduce the timeframe available for brute forcing (the attacker needs to break and use the password within a single shift) and they reduce the value of the stolen passwords because of its short time validity.

There are many password cracking software tools, but the most popular[37] are Aircrack-ng, Cain & Abel, John the Ripper, Hashcat, Hydra, DaveGrohl, and ElcomSoft. Many litigation support software packages also include password cracking functionality. Most of these packages employ a mixture of cracking strategies; algorithms with brute-force and dictionary attacks proving to be the most productive.[38]

The increased availability of computing power and beginner friendly automated password cracking software for a number of protection schemes has allowed the activity to be taken up by script kiddies.[39]

- Brute-force attack
- Cold boot attack
- Dictionary attack
- Password strength
- Smudge attack

1. ^ Jump up to: **a** **b** oclHashcat-lite – advanced password recovery Archived June 26, 2016, at the Wayback Machine. Hashcat.net. Retrieved on January 31, 2013.
2. ^ *Montoro, Massimiliano (2005). "Cain & Abel User Manual: Brute-Force Password Cracker". oxid.it (defunct). Archived from the original on June 7, 2019. Retrieved August 13, 2013.*
3. ^ *"What Is Password Spraying? How to Stop Password Spraying Attacks". Archived from the original on September 30, 2022. Retrieved September 16, 2021.*
4. ^ *Bahadursingh, Roman (January 19, 2020). "A Distributed Algorithm for Brute Force Password Cracking on n Processors". doi:10.5281/zenodo.3612276. Archived from the original on June 24, 2022. Retrieved June 24, 2022.*
5. ^ *Lundin, Leigh (August 11, 2013). "PINs and Passwords, Part 2". SleuthSayers.org. Orlando. Archived from the original on May 19, 2022. Retrieved June 24, 2022.*
6. ^ Alexander, Steven. (June 20, 2012) The Bug Charmer: How long should passwords be? Archived April 7, 2022, at the Wayback Machine. Bugcharmer.blogspot.com. Retrieved on January 31, 2013.
7. ^ Cryptohaze Blog: 154 Billion NTLM/sec on 10 hashes Archived March 6, 2016, at the Wayback Machine. Blog.cryptohaze.com (July 15, 2012). Retrieved on January 31, 2013.
8. ^ John the Ripper benchmarks Archived May 5, 2018, at the Wayback Machine. openwall.info (March 30, 2010). Retrieved on January 31, 2013.
9. ^ *Burr, W E; Dodson, D F; Polk, W T (2006). Electronic authentication guideline (PDF) (Report). Gaithersburg, MD: National Institute of Standards and Technology. doi:10.6028/nist.sp.800-63v1.0.2.*
10. ^ *"64-bit key project status". Distributed.net. Archived from the original on September 10, 2013. Retrieved March 27, 2008.*
11. ^ *"Password Recovery Speed table". ElcomSoft. Archived from the original on February 21, 2011. Retrieved February 1, 2011.*
12. ^ *"VCL Cluster Platform". mosix.cs.huji.ac.il. Archived from the original on October 19, 2021. Retrieved September 12, 2021.*
13. ^ *"25-GPU cluster cracks every standard Windows password in <6 hours". 2012. Archived from the original on January 27, 2019. Retrieved January 27, 2019.*
14. ^ *"EFF DES Cracker machine brings honesty to crypto debate". EFF. Archived from the original on January 1, 2010. Retrieved June 7, 2020.*
15. ^ *Biddle, Sam (May 11, 2017). "NYU Accidentally Exposed Military Code-breaking Computer Project to Entire Internet". The Intercept. Archived from the original on June 9, 2022. Retrieved June 6, 2020.*
16. ^ *"announce - [openwall-announce] John the Ripper 1.9.0-jumbo-1". openwall.com. Archived from the original on November 8, 2020. Retrieved June 6, 2020.*
17. ^ *"Bcrypt password cracking extremely slow? Not if you are using". Medium. September 8, 2020. Archived from the original on September 20, 2021. Retrieved September 12, 2021.*
18. ^ Managing Network Security. Fred Cohen & Associates. All.net. Retrieved on January 31, 2013.
19. ^ *Yan, J.; Blackwell, A.; Anderson, R.; Grant, A. (2004). "Password Memorability and Security: Empirical Results" (PDF). IEEE Security & Privacy. **2** (5): 25. doi:10.1109/MSP.2004.81. S2CID 206485325. Archived (PDF) from the original on December 6, 2022. Retrieved June 24, 2022.*
20. ^ Jump up to: **a** **b** *Steinberg, Joseph (April 21, 2015). "New Technology Cracks 'Strong' Passwords – What You Need To Know". Forbes. Archived from the original on August 31, 2017. Retrieved September 7, 2017.*
21. ^ *"CERT IN-98.03". Archived from the original on July 9, 2010. Retrieved September 9, 2009.*

22. ^ Jump up to: ***a*** ***b*** *"Consumer Password Worst Practices" (PDF). Imperva.com. Archived (PDF) from the original on June 24, 2022. Retrieved June 24, 2022.*
23. ^ *"NATO Hack Attack". The Register. Archived from the original on July 26, 2020. Retrieved July 24, 2011.*
24. ^ *"Anonymous Leaks 90,000 Military Email Accounts in Latest Antisec Attack". July 11, 2011. Archived from the original on June 24, 2022. Retrieved June 24, 2022.*
25. ^ *"Military Password Analysis". Imperva.com. July 12, 2011.*
26. ^ *"Microsoft's Hotmail Bans 123456". Imperva.com. July 18, 2011. Archived from the original on March 27, 2012.*
27. ^ *"Ashley Madison: Hackers Dump Stolen Dating Site Data". bankinfosecurity.com. Archived from the original on April 11, 2021. Retrieved April 11, 2021.*
28. ^ *"Researchers Crack 11 Million Ashley Madison Passwords". bankinfosecurity.com. Archived from the original on April 11, 2021. Retrieved April 11, 2021.*
29. ^ *Singer, Abe (November 2001). "No Plaintext Passwords" (PDF). Login. **26** (7): 83–91. Archived (PDF) from the original on September 24, 2006.*
30. ^ *"Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol". Schneier.com. July 7, 2011. Archived from the original on May 21, 2016. Retrieved January 31, 2013.*
31. ^ *Grassi, Paul A (June 2017). "SP 800-63B-3 – Digital Identity Guidelines: Authentication and Lifecycle Management" (PDF). NIST. doi:10.6028/NIST.SP.800-63b. Archived (PDF) from the original on April 1, 2019. Retrieved February 3, 2021.*
32. ^ A Future-Adaptable Password Scheme Archived September 24, 2008, at the Wayback Machine. Usenix.org (March 13, 2002). Retrieved on January 31, 2013.
33. ^ MDCrack FAQ 1.8 Archived August 27, 2008, at the Wayback Machine. None. Retrieved on January 31, 2013.
34. ^ Password Protection for Modern Operating Systems Archived March 11, 2016, at the Wayback Machine. Usenix.org. Retrieved on January 31, 2013.
35. ^ *"Password Hashing Competition". Archived from the original on September 2, 2013. Retrieved March 3, 2013.*
36. ^ *"NIST SP800-63B Section 5.1.1.2" (PDF). nvlpubs.nist.gov. Archived (PDF) from the original on April 1, 2019. Retrieved February 3, 2021.*
37. ^ *"Top 10 Password Crackers". Sectools. Archived from the original on May 11, 2019. Retrieved November 1, 2009.*
38. ^ *"Stay Secure: See How Password Crackers Work - Keeper Blog". Keeper Security Blog - Cybersecurity News & Product Updates. September 28, 2016. Archived from the original on October 21, 2020. Retrieved November 7, 2020.*
39. ^ *Anderson, Nate (March 24, 2013). "How I became a password cracker: Cracking passwords is officially a "script kiddie" activity now". Ars Technica. Archived from the original on October 26, 2017. Retrieved March 24, 2013.*

- Philippe Oechslin: Making a Faster Cryptanalytic Time-Memory Trade-Off. Archived April 9, 2023, at the Wayback Machine CRYPTO 2003: pp617–630
- Roundup of leaks made by The Anonymous and LulzSec in 2011
- International passwords conference

- Password security: past, present, future, Passwords12 presentation
- Skullsecurity list of breached password collections Archived December 15, 2019, at the Wayback Machine

---