Passwords – guessing and brute-forcing

Martin Stanek

2024

Table of Contents

Motivation and introduction

- passwords are still ubiquitous authentication mechanism
  - MFA highly recommended
- stolen passwords – (spear)phishing, browsers, configuration files, logs, backups, etc.
- passwords stored/transmitted in clear – easy
- passwords stored as a hash or encrypted, examples:
  - `/etc/shadow` (Linux)
  - `SAM` Security Accounts Manager, together with `SYSTEM` (Windows)
  - `LSASS` Local Security Authority Subsystem Service (Windows)
  - web application accounts
- passwords used in an authentication protocol:
  - offline dictionary attack (using eavesdropped communication) – WPA2, NTLMv2
  - solution: PAKE (Password-authenticated key exchange) – SRP, SAE/Dragonfly
  - online attack always possible (other countermeasures to make it hard/infeasible)

Protocol example – Net-NTLMv2

- password hash (LM:NT), no salt, both are weak:
  - LM hash – LanManager hash, disabled in recent Windows versions
  - NT hash – `MD4(UTF-16-LE(password))`
- NTLMv2 (Net-NTLMv2) is a challenge/response authentication protocol
  - proving knowledge of the password, more precisely the knowledge of the hash
  - simplified, authentication check with domain controller, etc.
  - dictionary or brute-force attack after obtaining the response

Protocol example – Net-NTLMv2 (2)

1. Client → Server: initiate connection, `user name`
2. Server → Client: options, challenge `SC` (8-byte random value)
3. Client → Server: response: `LMv2, client nonce 1, NTv2, NT-state`

```
v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)
NT-state = (fixed data, current time, client nonce 2, domain name)
LMv2 = HMAC-MD5(v2-Hash, SC, client nonce 1)
NTv2 = HMAC-MD5(v2-Hash, SC, NT-state)
```

Brute force speed

- two hash functions:
  - simple SHA256
  - sha512crypt 6, SHA512 (Unix) [Iterations: 5000]

platform	SHA256	sha512crypt
Apple M3 Pro, GPU	1088 MH/s	
NVIDIA GeForce RTX 4090	21976 MH/s	1180 kH/s
NVIDIA H100	12200 MH/s	666 kH/s

benchmarks: https://github.com/Chick3nman

Datasets

- Crackstation.net
  - MD5 and SHA1 hashes: 15-billion-entry lookup table (190 GB)
  - other hashes (LM, NTLM, sha256, ...): 1.5-billion-entry lookup table (19 GB)
- SecLists project (Password directory)
  - leaked, captured, cracked, default passwords
  - RockYou and others
- "Largest password leaks"
  - 2017 Breach Compilation, 1.4 billion email and password pairs
  - 2021 Compilation of Many Breaches, 3.2 billion unique pairs
  - 2024 another compilation ("Mother of all breaches")

Tools – John the Ripper, Hashcat

- CPU and GPU password cracking
- some Hashcat's attack modes:
  - *Straight* – dictionary attack
  - *Combination* – concatenating words from exactly two dictionaries
  - *Brute-force* / *Mask* – prescribe character sets
  - *Hybrid Wordlist and Mask* – combination of these two attacks
  - *Rule based* – applying various transformations (uppercase, append character, etc.)
- mask examples:
  - `-a 3 ?u?l?l?l` – 4-letter strings with first uppercase character (`Aaaa` – `Zzzz`)
  - `-a 3 password?d?d` – corresponding keyspace: `password00` – `password99`

Custom dictionaries and rules

- generic password lists might not be enough
- tailor dictionary for the target (add to or replace generic lists)
  - CeWL (Custom Word List generator) – words from a web site
  - other OSINT tools and techniques
  - using any a priori knowledge of the target
  - extend dictionary with new data as testing progresses
- rules: password modifiers – derive new passwords
  - append, capitalize the first letter, reverse, delete, replace etc.
  - example: `s _ $!` replace all spaces with _ and append !
- popular rule sets:
  - base66.rules (see hashcat's `rules` directory for more)
  - OneRuleToRuleThemAllStill (focus mostly on cracking fast hashes)

Resources

1. Joshua Picolet: *Hash Crack: Password Cracking Manual (v3)*, 2019
2. Awesome Password Cracking