



SP2025 Week 06 • 2025-03-06

# Password Cracking

Adarsh Krishnan and Louis Asanaka

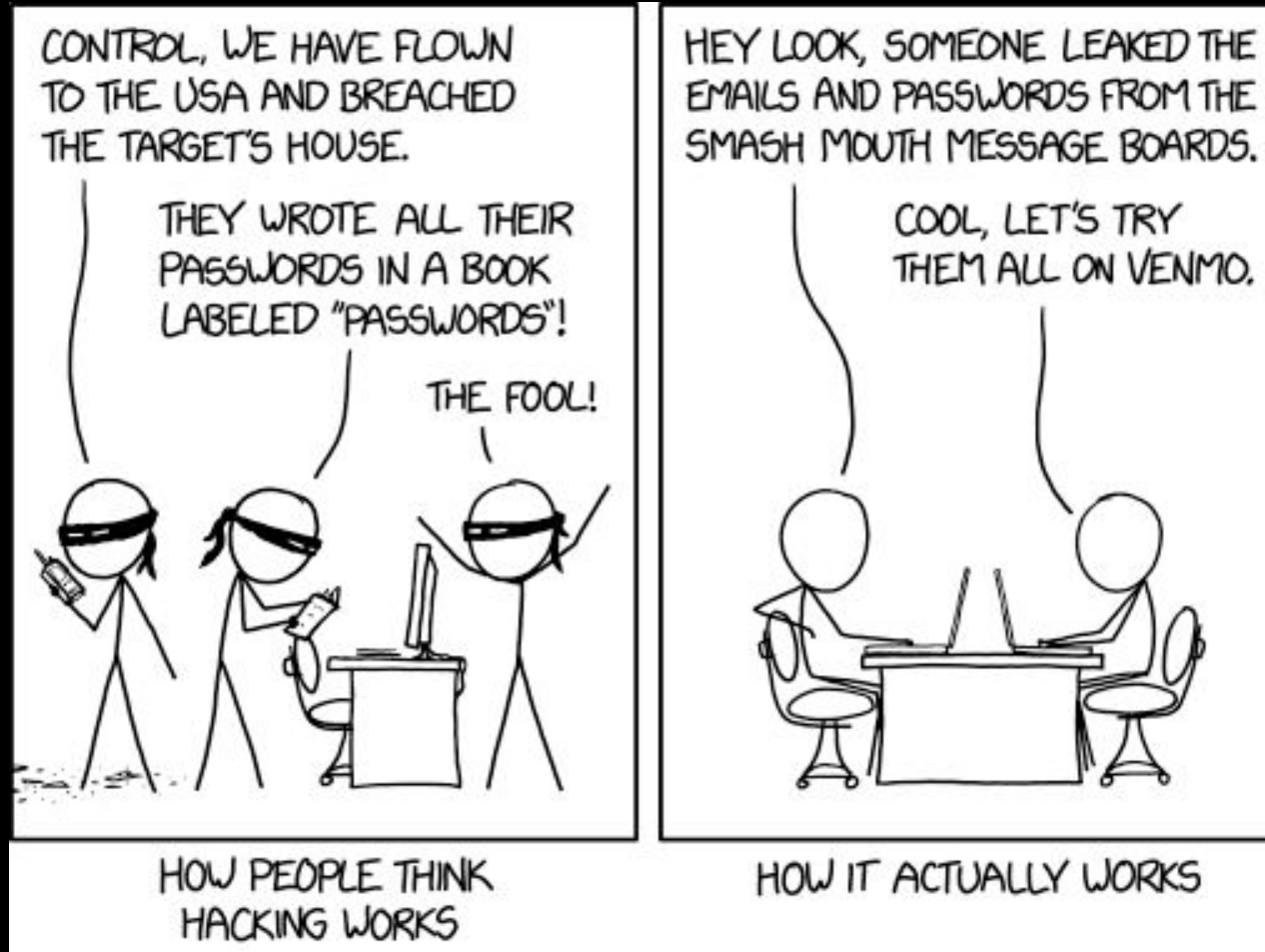
# Announcements

- LAN Party next week
  - Current idea is Minecraft server
  - Share any ideas for other games if you want



ctf.sigpwny.com

sigpwny{H4\$h\_Br0wn5}



# What is Hashing?

- A one way function (irreversible!) that takes something in plaintext and generates a garbled, fixed-length collection of bytes
- Every plaintext has its unique hash- we sometimes call it a fingerprint

Plaintext: "Hi"

MD5 hash: c1a5298f939e87e8f962a5edfc206918

Plaintext: "Hii" (example of the avalanche effect)

MD5 hash: 65c1aa5487711a23f7477200fd01e253

Plaintext: "Wow, this is a very long sentence!"

MD5 hash: 0df95e5a7e96079e67663243e29aeba3



# Hashing Passwords

- Never store plaintext; store their hash instead
- When you login with a password, the hash of the password is calculated and checked with the stored correct hash
- This way if an attacker obtains a hash, they can't obtain user passwords unless they brute force them

Server stores:	5f4dcc3b5aa765d61d8327deb882cf99
Client Guess 1: hunter2	2ab96390c7dbe3439de74d0c9b0b1767
Client Guess 2: password	5f4dcc3b5aa765d61d8327deb882cf99

# Cracking Password Hashes

- As mentioned earlier, hashing is one way, so you can't easily get a password if you're given a hash
- The only way is to brute force or "crack" the hash with guesses of what the password could be
  - **Pure brute force**: using every possible combinations of letters, numbers, and symbols (takes a LONG time)
  - **Dictionary/wordlist attack**: limits the amount of guesses by guessing common, or likely valid passwords
  - **Rule-based dictionary attack**: like a dictionary attack but more versatile



# Computing Hashes

- Generating a single hash takes less than a second but computing a lot of hashes in a brute force attack takes much longer
- Highly dependent on your compute power
- GPUs can crack passwords faster than CPUs
- People build custom password cracking rigs with lots of GPUs or just rent a high-compute cloud server on AWS/Google Cloud/Azure



# Popular Wordlists

- rockyou.txt
  - This is the standard wordlist to use if you don't know anything about the password you're cracking
  - Contains ~32 million common passwords sourced from various password breaches
  - Short enough to go through in a reasonable amount of time, comprehensive enough for most cases
  - Most CTFs will use passwords from rockyou to simplify password cracking and make challenges not impossible
- SecLists
  - Repository of even more prebuilt wordlists





# Creating a Custom Wordlist

- A wordlist attack has a higher chance of succeeding if you use words/phrases related to whoever created the password, instead of a generic wordlist like rockyou
  - e.g. pet names, song lyrics, keywords
- Tools: Mentalist, hashcat rules/hybrid attack



# Different Hashing Functions

- MD5
- SHA256
- SHA512 (newer Linux systems)
- md5crypt (older Linux systems)
- NTLMv2 (newer Windows systems)
- LM (ancient Windows systems)
- RIPEMD (RIPE project from the EU)



# Not All Hashes Are Created Equal

- A hashing function that takes more time to compute makes password cracking less effective
  - MD5 takes around 500ms to compute a hash
  - SHA512 takes around 1000ms to compute a hash (double of MD5)
  - Verifying a password takes double time (still insignificant), but cracking the password takes exponentially more time
- Bigger hashes sizes reduces the possibility of hash collisions
  - MD5 hashes contain 16 bytes
  - SHA512 hashes contain 64 bytes



# Salting

- Adds a random string to the password before hashing
- This way an attacker can't just precompute all possible hashes beforehand and compare a hash to the database of all hashes
  - These precomputed databases are called rainbow tables

Salt = 59c^ad

Password = Password@123

What's hashed: 59c^adPassword@123

- The salt is stored with the hash



# A Lesson in Hashes (ft. Microsoft)

LM hash function steps:

1. Convert all lowercase to uppercase
2. Pad password to 14 characters with NULL characters (or get rid of extra characters after first 14 characters)
3. Split the password to two 7 character chunks
4. Create two DES keys from each 7 character chunk
5. DES encrypt the string "KGS!@#\$\$%" with these two chunks
6. Concatenate the two DES encrypted strings. This is the LM hash.



# A Lesson in Hashes (ft. Microsoft)

Original password: R3@1LyS3cur3P4sSw0rd!8953

Effective password: R3@LLYS3CLR3P4 (limited to 14 characters, uppercase only)

Split into two hashes:

R3@LLYS -> 0081A58503BD7E1D

3CLR3P4 -> 5438E4541FAB2BDE

Full LM hash: 0081A58503BD7E1D5438E4541FAB2BDE

You can just precompute every 7 character password combination and compare the hash!



# Tools



# Hashcat

- Hashcat is an extremely fast (read: uses GPU) "password recovery" tool, which automates cracking passwords using the methods described earlier.
  - Three main command line switches you should know about:
    - -m NUM: Chooses the hash type, that is, which kind of hash you're trying to "reverse." For instance, `-m 0` is MD5.
    - -a NUM: Chooses the attack mode. Most of our challenges will use 0, straight mode.
    - -h: Help. Get a list of all hash types, attack modes, etc. supported by Hashcat.
  - See [https://hashcat.net/wiki/doku.php?id=example\\_hashes](https://hashcat.net/wiki/doku.php?id=example_hashes) to figure out what hash mode to use for the -m flag





# Hashcat Example

- Run Hashcat on example400.hash, using example.dict as a wordlist
  - `hashcat -a 0 -m 400 example400.hash example.dict`
- Same as above, but using Hashcat "rules":
  - `hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule`
- A Hashcat rule is a sort-of regular expression to try variants on your wordlist
- Hashcat has brute-force/hybrid attacks that are useful

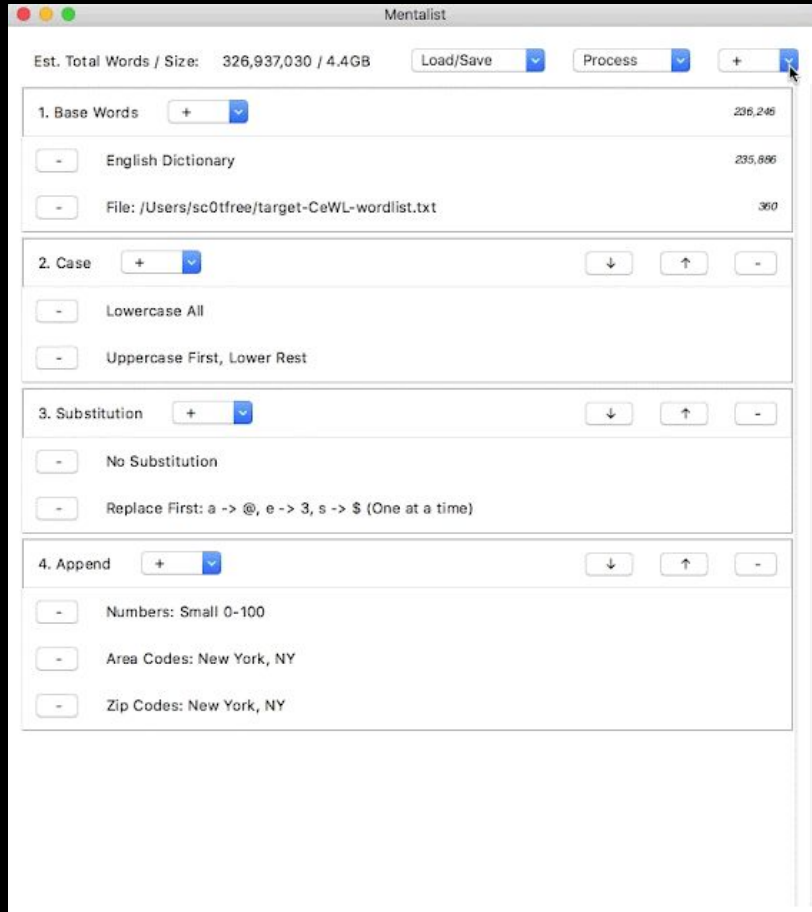


# John the Ripper

- Similar to hashcat, we recommend hashcat though
- Useful JtR exclusive tools (can download separately as a python script and feed into hashcat)
  - zip2john to create a password hash from an encrypted zip archive
  - pdf2john to create a password hash from an encrypted PDF file
- Use Hashcat's example hashes to identify the hash mode to use for the hash you are trying to crack



# Mentalist



Easy to use GUI program to prepend or append numbers, characters, or even other wordlists to a wordlist!



# PDFrip

```
█
```

- Multithreaded CLI tool designed to crack passcode-locked PDFs



# Resources

- Hashcat: <https://hashcat.net/hashcat/>
- rockyou.txt:  
<https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt>
- Mentalist: <https://github.com/sc0tfree/mentalist>
- SecLists:  
<https://github.com/danielmiessler/SecLists/tree/master/Passwords>
- PDFrip: <https://github.com/mufeedvh/pdfrip>



# Next Meetings

## 2025-03-09 • This Sunday

- PWN IV: Heap Exploitation with Nikhil and Akhil
- Learn about modern binary exploitation techniques

## 2025-03-13 • Next Thursday

- LAN Party!
- Share any ideas if you have them

## 2025-03-16 • Next Sunday

- No meeting, have a good break!



ctf.sigpwny.com

sigpwny{H4\$h\_Br0wn5}

Meeting content can be found at  
[sigpwny.com/meetings](https://sigpwny.com/meetings).

