Password Security

Need for passwords

- Passwords represent a simple authentication mechanism
  - Something you know
- Despite efforts to be replaced by other methods, passwords remain a popular way to implement authentication for web and mobile systems

Securing passwords

- Depends on the security model, the application, and the usability requirements
- Security of passwords depends on password creation **policies**.
  - Passwords can be simple whenever cracking it is not a critical threat.
- Who should or should not see the passwords?

Attacks to crack passwords

- Brute force attacks
- Probabilistic attacks
- Dictionary attacks

Brute force attacks

- Brute force methods assume little knowledge about the passwords' composition.
- Brute force methods tend to try the entire *password space*. They hope to find it soon!
- Mostly require offline attacks since online attacks can be easily prevented with **trial limits**.

Password length plays a role

26 UPPER/lower case characters = 52 characters
10 numbers
32 special characters
=> 94 characters available

5 characters: 94^5 =	7,339,040,224
6 characters: 94^6 =	689,869,781,056
7 characters: 94^7 =	64,847,759,419,264
8 characters: 94^8 =	6,095,689,385,410,816
9 characters: 94^9 =	572,994,802,228,616,704

Passwords often subject to rules

- Password update policies
  - Mandatory policies, optional policies, and a combination
- Password length policies
  - Mandatory
- Password composition rules
  - Mandatory

How long does it take?

Example Rule: Password does not change for 60 days
how many passwords should I try for each second?

5 characters:	1,415 PW /sec
6 characters:	133,076 PW /sec
7 characters:	12,509,214 PW /sec
8 characters:	1,175,866,008 PW /sec
9 characters:	110,531,404,750 PW /sec

Exercise: How many passwords can you check on your machine per second?

Mask brute force attacks

- Directed brute force attacks that use password policies to cut the search space.
- Many users choose passwords with structural patterns. This gives attackers a chance to make better guesses.
- Example:
- Password must contain digits
- Actual password: noonebeatsme1985
  - Rule out all guesses that lead to “noonebeatsme” alone.

Dictionary attacks

- Wordlists used to try a large number of candidate passwords
- Wordlists contain dictionary words and previously stolen passwords
- Mangling rules perform transformations on wordlist words
  - Example: word: monkey, mangling rule: append 99. Try: monkey99

Dictionary attacks

- Also used to perform password recovery
- Often more guided with a shorter more likely subset of the wordlist
- Mangling rules frequently updated manually by experts
- Example publicly available wordlists
  - https://wiki.skullsecurity.org/Passwords

Guided attacks

- Actual password: noonebeatsme1985
  - has words “no” “one” “beats” “me” “1985”
- Mask examples:
  - When trying “one”: “noone” and “oneno”
  - When trying “noone”: “nooneeats” “noonekills” …
  - When trying “noonebeatsme”: “2000”+”noonebeatsme” …
- With manual guidance may lead to reasonable results

Password Policies

- Minimum char. classes & minimum-length requirement
  - strengthen passwords without decreasing their memorability
  - but insufficient to protect against effective attacks
- Example:
  - 4-class, 8-character
  - 1-class, 16-character
- Password strengthening technique: blocklist, symbol-stripping

Password Block List

- Databases: Xato, Pawned API
- Possible block list policies:
  - case-insensitive full-string (cifs);
  - case-sensitive full-string (fs);
  - stripping digits and symbols and then performing a case-insensitive full-string comparison (strip-cifs); and
  - checking whether any length-5 substring of any wordlist entry was a case-insensitive substring of the candidate password (ciss)

Password Guess Number

- Number of trials needed for a password cracker to guess the password.
- Famous crackers: Hashcat & John the Ripper
- Using Probabilistic Context-Free Grammars (PCFG) to reason about the guessability of a password.

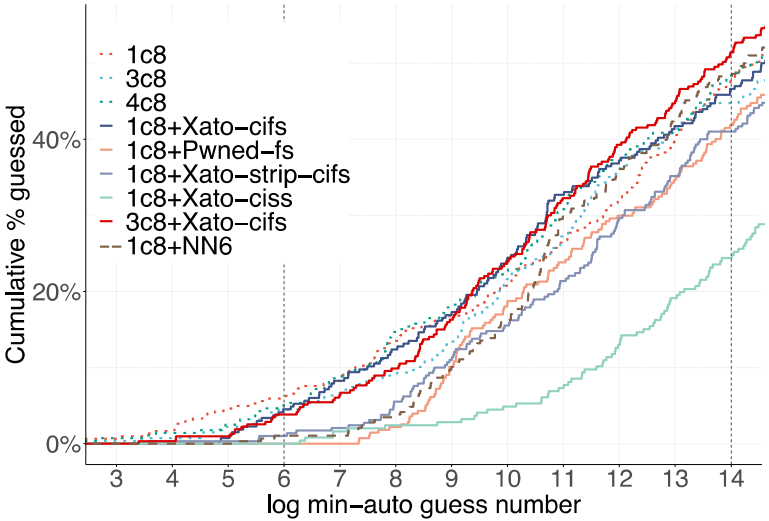
Probabilistic attacks

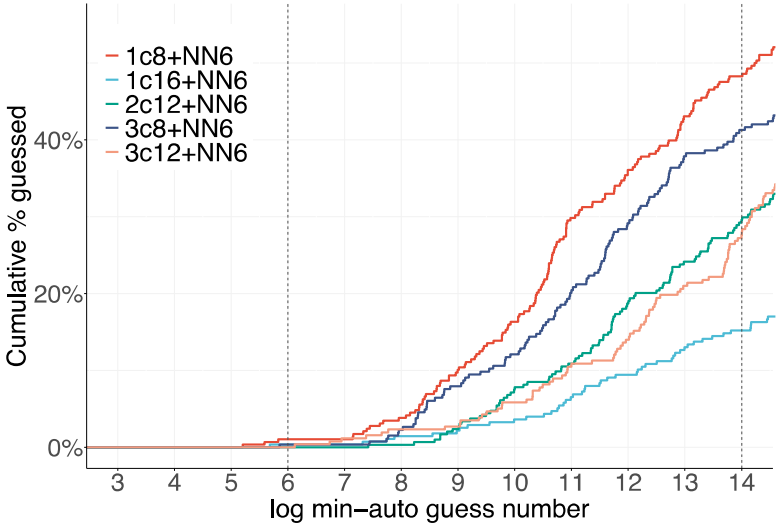
- Probabilistic context-free grammars
  - Use a set of training data to assign probabilities to password structures, components, etc.
- Differentiate upper and lowercase letters
- Assign probabilities to unseen structures
- Probabilities make sense in controlled experiments

Probabilistic attacks

- Example
  - Probability of finding word “hacker” in the passwords of all CS students at Kuwait University.
  - Probability of finding a common Kuwaiti jargon in the passwords of all Kuwait University students.
- Elements to consider: the entire data set, the sample sets, the criteria to consider

Some results





Reference: Tan et al., CCS'20

Passwords in software

- Responsibility of software engineers to help securing passwords
- Passwords must be securely stored
  - Example: using a cryptographic hash function. Never use plaintext passwords.
  - Do not use passwords embedded in source code that is distributed to clients.

Hashing will not prevent brute force attacks

- Hashing the password may not be enough.
- Consider an attacker that has a powerful machine and can try many passwords from the dictionary in a very short time.
- Also, the attacker will be able to quickly produce the hashed password (assuming the attacker knows the hashing algorithm) and give a try.
- One key defense is to slow down the attacker.

Lookup tables

- Adversaries may also attack passwords by using lookup tables.
- A table that has passwords and their corresponding hashes. A brute force attack is used to reverse the hashing.

Password	Hash	Algorithm
noonebeatsme195	4e74f73caa229197a168bd0e9727d04f45ba4e9c	SHA1
iliveinkw	d2e95ea4ce3d3a04ac7063f0c3437fa0e6d17b4b	SHA1
80949999	098f6908c334d627bb43908d6274e57b21e99ce1	SHA1
skyismy99limit	3ff131e9e052192acac15d9fcd672639b273b9e2	SHA1

Password stretching

- A cryptographic technique used to slow down a brute force (or dictionary) attack on hashed passwords
- The idea is to use a cryptographic hash function in a large number of iterations (e.g., 100,000 iterations) applied to the original plaintext password.
- The higher the number of iterations, the slower the attack

Password salting

- Adding random values to the hashing function to prevent dictionary and brute force attacks on passwords.
- Actual password: noonebeatsme, salt: 03990855xi
- Hashing: h(salt,password) → hash

More Types of Passwords

- Image-based authentication
  - Collective vs. individual educated guess attacks
- Drawing-based authentication (recall-based)
  - Users tend to use symmetry and small strokes
- Cued recall methods
  - Selecting regions or points (recall) on a graph (cue)
  - People choose predictable hot spots
- Recognition methods (have you seen this before?)
  - Users are likely to choose images of females from their ethnic groups
  - Image distortion can work with user-selected images