



CSE 127: Computer Security

# Password Authentication

Kirill Levchenko

November 7, 2017

# The Problem

- ❖ Security policies refer to persons
  - Legal deterrents apply to persons
  - Systems built for people not machines
- ❖ People don't exist at machine level
- ❖ How do we attribute actions to persons?

# Identity

- ❖ **Identity:** the totality of characteristics that define an object in a particular context
- ❖ **Identifier:** a name uniquely determining an object in a particular context
- ❖ We use identifiers to name objects in a system
  - *Objects includes subjects*

# Identity Example

- ❖ **Context:** California Department of Motor Vehicles
- ❖ **Identifier:** driver license number
- ❖ **Identity:** the legal person

# Authentication Goal

- ❖ Determine *who* is interacting with the system
  - Necessary to apply appropriate security policy
- ❖ ***Authenticate:*** prove you are who you say you are
- ❖ ***Security property:*** only the intended subject should be able authenticate to the system as that subject
  - If a less privileged subject acts as more privileged subject, security policy may be violated

# Authentication Kinds

- ❖ Something you know
  - Password, other identifiers
- ❖ Something you have
  - Card, key, one-time password device
- ❖ Something you are
  - Biometrics

# Authentication

- ❖ Many authentication schemes have two elements:
  - Provide an identifier to the system
  - Prove that you are the subject associated with identifier
  - **Example:** user name and password

# Popular Mechanisms

- ❖ Username and password
  - Two factor: secondary verification, e.g., via phone
- ❖ Card and PIN
  - No need to explicitly enter identifier
- ❖ Biometric
  - E.g. fingerprint, voice, face, retina scan

# Passwords

- ❖ Password: Secret known only to subject that allows subject to authenticate
  - “Something you know”
- ❖ Need a mechanism to prove to system that subject knows the password

# Password Authentication

- ❖ Simplest: subjects gives the password to system
  - May be observed by passive adversary:  
Need communication channel to protect confidentiality
  - Attacker may impersonate system:  
Need to authenticate system!
- ❖ One-time-password: subject gives system a one-time use secret derived from master password

# User-Side Compromise

- ❖ **Old school:** post-it note with password stuck to monitor
  - Targeted attacks against particular user
- ❖ **Modern:** electronic capture during entry
  - Insecure Internet connection
    - Man-in-the-middle
    - Unencrypted
  - Keystroke logger
    - Targeted attack
    - Generic malware

# System-Side Compromise

- ❖ System may be compromised and passwords stolen
  - Huge yield compared to user-side attacks
- ❖ Defenses:
  - Protect the password database
  - Make password recovery from database difficult

# Checking Password

- ❖ System does not need to *know* password, only *check* it
- ❖ **One-way function:** easy to compute, hard to invert
  - Also called pre-image resistance
- ❖ **Solution:** store output of function on password:  $h = \text{owf}(p)$ 
  - Vulnerable to dictionary attacks (discussed next)
  - Pre-image resistance implies hard to get password

# Dictionary Attack

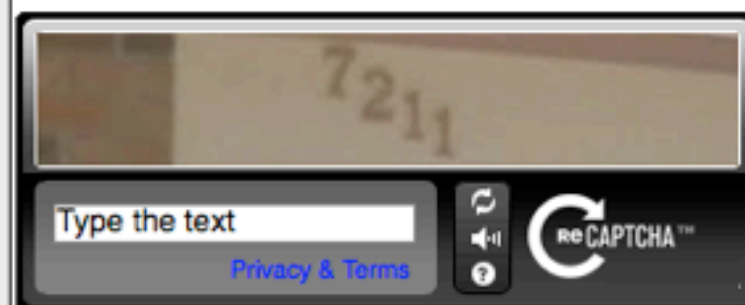
- ❖ Attacker generates hashes of all dictionary words and/or all strings up to certain length
- ❖ **Example:**
  - Each password character upper or lower case letter or digit
  - Roughly 64 possible values per character
  - $64^n$  possible passwords of length  $n$
  - For  $n = 6$ :  $2^{36}$  possible passwords strings:
    - 10TB to store 6-character password and SHA1 hash

# Dictionary Attack

- ❖ **Dictionary:** list of all possible passwords of certain form and their hashes
- ❖ Compare all possible passwords against password list
  - For  $n = 6$ : 10TB sort—feasible
- ❖ Very common attack today

## Free Password Hash Cracker

Enter up to 10 non-salted hashes:



Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5), md5-half, sha1, sha1(sha1\_bin()), sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+

## [Download CrackStation's Wordlist](#)

### How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with

# Dictionary Attack

- ❖ What is the problem? Work hashing one possible password (when generating table) amortized over all password we compare against
- ❖ Total work  $O(m+n)$ 
  - $m$  — size of dictionary table
  - $n$  — size of password database trying to crack
- ❖ Want:  $O(m \times n)$ 
  - Have to brute force every password

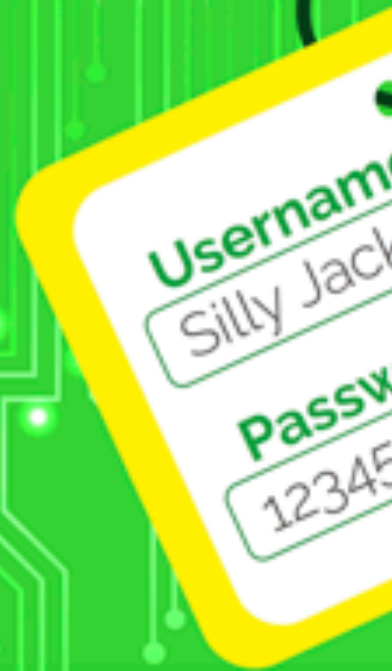
# Salting

- ❖ Salting: add additional random value to password
- ❖ Generate random salt  $s$
- ❖ Store  $\langle s, \text{MAC}_p(s) \rangle$  in the database
  - Additional iterations added to slow down brute forcing
- ❖ Attacker would need dictionary tables for every salt
  - Large salt space makes this infeasible
- ❖ Can attack each password individually
  - Does not parallelize

# Brute Forcing

- ❖ **Simplest idea: try every password**
  - Works against salted table
- ❖ **Better: try common passwords first**
  - Dictionary words, common passwords
- ❖ **Countermeasure: key stretching (iterated hashing)**

# WORST PASSWORDS OF 2015



**SplashData** releases its annual list in an effort to encourage the adoption of stronger passwords to improve Internet security. The passwords evaluated are mostly from North American and Western European users. The list shows many **people continue to put themselves at risk for hacking and identity theft** by using weak, easily guessable passwords.

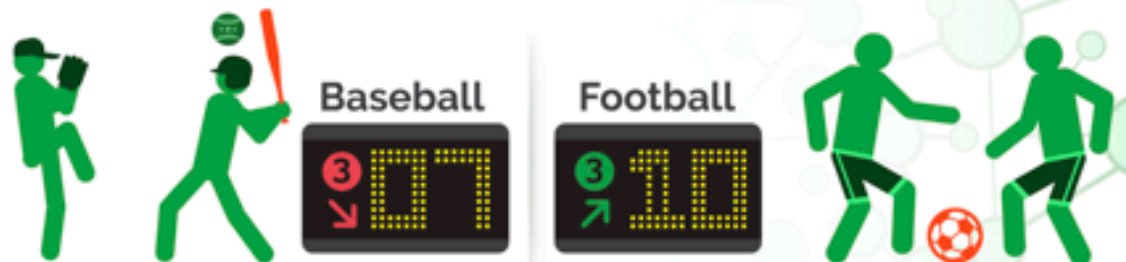
RANK	PASSWORD	CHANGE FROM 2014
1	123456	Unchanged
2	password	Unchanged
3	12345678	1 ↗
4	qwerty	1 ↗
5	12345	2 ↘
6	123456789	Unchanged
7	football	3 ↗
8	1234	1 ↘
9	1234567	2 ↗
10	baseball	2 ↘
11	welcome	NEW
12	1234567890	NEW
13	abc123	1 ↗
14	111111	1 ↗
15	1qaz2wsx	NEW
16	dragon	7 ↘



"123456" and "password" once again reign supreme as the most commonly used passwords



Some longer passwords are so simple as to make their extra length virtually worthless



# Practical Usage

- ❖ Many libraries have password hashing functions
- ❖ E.g. PHP's `password_hash`
- ❖ Incorporate salt and key stretching