User Authentication and Passwords

CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 28 October 2013

css322y13s2l11, Steve/Courses/2013/s2/css322/lectures/passwords.tex, r2963

Contents

User Authentication

Password-Based Authentication

Password Entropy

Storing Passwords

Selecting Passwords

User Authentication

The process of verifying a claim that a system entity or system resource has a certain attribute value.

— R. Shirey, “Internet Security Glossary, Version 2”, IETF RFC4949

Two Steps of Authentication

1. Identification step: presenting an identifier to the security system
   - E.g. user ID
   - Generally unique but not secret
2. Verification step: presenting or generating authentication information that acts as evidence to prove the binding between the attribute and that for which it is claimed.
   - E.g. password, PIN, biometric information
   - Often secret or cannot be generated by others

User authentication is primary line of defence in computer security; other security controls rely on user authentication

Means of Authentication

Something the individual . . .

Knows

- E.g. password, PIN, question answers

Possesses

- Token, e.g. keycards, smart card, physical key

Is

- Static biometrics, e.g. fingerprint, retina, face

Does

- Dynamic biometrics, e.g. voice pattern, handwriting, typing rhythm

Humans and Computers

Humans are also large, expensive to maintain, difficult to manage and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.

— Kaufman, Perlman, Speciner "Network Security: Private Communication in a Public World", Prentice Hall 2002

Contents

User Authentication

Password-Based Authentication

Password Entropy

Storing Passwords

Selecting Passwords

Password-Based Authentication

- Many multiuser computer systems used combination of ID and password for user authentication
- System initially stores username and password
- User submits username/password to system; compared against stored values; if match, user is authenticated
- Identity (ID):
  - Determines whether user us authorised to gain access to system
  - Determines privileges of user, e.g. normal or superuser
  - Used in access control to grant permissions to resources for user
- Password:
  - What is a good password?
  - How to store the passwords?
  - How to submit the passwords?
  - How to respond (if no match)?

Vulnerability of Passwords

Offline Dictionary Attack Attacker obtains access to ID/password (hash) database; use dictionary to find passwords

- Countermeasures: control access to database; reissue passwords if compromised; strong hashes and salts

Specific Account Attack Attacker submits password guesses on specific account

- Countermeasure: lock account after too many failed attempts

Popular Password Attack Try popular password with many IDs

- Countermeasures: control password selection; block computers that make multiple attempts

Vulnerability of Passwords

Password Guessing Against Single User Gain knowledge about user and use that to guess password

- Countermeasures: control password selection; train users in password selection

Computer Hijacking Attackers gains access to computer that user currently logged in to

- Countermeasure: auto-logout

Exploiting User Mistakes Users write down password, share with friends, tricked into revealing passwords, use pre-configured passwords

- Countermeasures: user training, passwords plus other authentication

Vulnerability of Passwords

Exploiting Multiple Password Use Passwords re-used across different systems/accounts, make easier for attacker to access resources once one password discovered

- Countermeasure: control selection of passwords on multiple account/devices

Electronic Monitoring Attacker intercepts passwords sent across network

- Countermeasure: encrypt communications that send passwords

Contents

User Authentication

Password-Based Authentication

Password Entropy

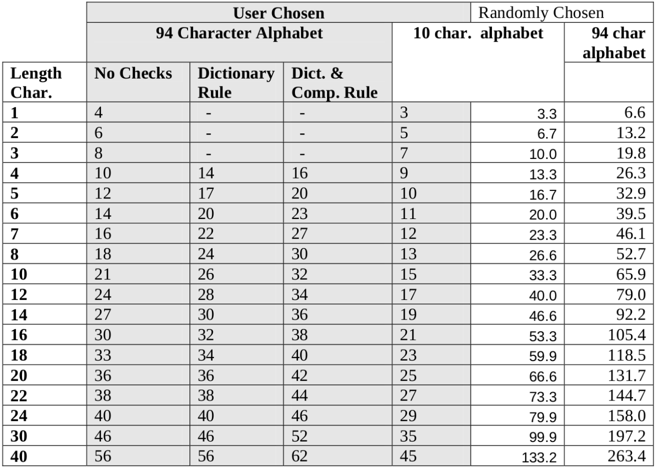
Storing Passwords

Selecting Passwords

Strength of Passwords

- Entropy used as indicator of password strength
  - Password with entropy of n bits is equivalent to n-bit key at withstanding brute force
  - How many bits needed to represent symbols from symbol set:
    - Digits, 0 . . . 9: 3.32
    - English letters, a . . . z: 4.70
    - Printable ASCII characters (94): 6.55
  - For 64-bit equivalent strength:
    - Digits: 20
    - English letters: 14
    - Printable ASCII characters: 10
- Human generated passwords are not random
  - Difficult to estimate entropy, NIST have approximations

NIST Estimated Password Strength



	User Chosen				Randomly Chosen	
	94 Character Alphabet			10 char. alphabet		94 char alphabet
Length Char.	**No Checks**	**Dictionary Rule**	**Dict. & Comp. Rule**			
1	4	-	-	3	3.3	6.6
2	6	-	-	5	6.7	13.2
3	8	-	-	7	10.0	19.8
4	10	14	16	9	13.3	26.3
5	12	17	20	10	16.7	32.9
6	14	20	23	11	20.0	39.5
7	16	22	27	12	23.3	46.1
8	18	24	30	13	26.6	52.7
10	21	26	32	15	33.3	65.9
12	24	28	34	17	40.0	79.0
14	27	30	36	19	46.6	92.2
16	30	32	38	21	53.3	105.4
18	33	34	40	23	59.9	118.5
20	36	36	42	25	66.6	131.7
22	38	38	44	27	73.3	144.7
24	40	40	46	29	79.9	158.0
30	46	46	52	35	99.9	197.2
40	56	56	62	45	133.2	263.4

NIST Special Publication 800-63, Electronic Authentication Guideline, April 2006. `http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf`

Contents

User Authentication

Password-Based Authentication

Password Entropy

Storing Passwords

Selecting Passwords

Storing Passwords

- Upon initial usage, user ID and password are registered with system
- ID, password (or information based on it), and optionally other user information stored on system, e.g. in file or database
- To access system, user submits ID and password, compared against stored values
- How should passwords be stored?

Storing Passwords in the Clear

$$ID, P$$

Insider attack: normal user reads the database and learns other users passwords

- Countermeasure: access control on password database

Insider attack: admin user reads the database and learns other users passwords

- Countermeasure: none—admin users must be trusted!

Outsider attack: attacker gains unauthorised access to database and learns all passwords

- Countermeasure: do not store passwords in the clear

Encrypting the Passwords

$$ID, E(K, P)$$

- Encrypted passwords are stored
- When user submits password, it is encrypted and compared to the stored value
- Drawback: Secret key, K, must be stored (on file or memory); if attacker can read database, then likely they can also read K

Hashing the Passwords

$$ID, H(P)$$

- Hashes of passwords are stored
- When user submits password, it is hashed and compared to the stored value
- Practical properties of hash functions:
    - Variable sized input; produce a fixed length, small output
    - No collisions
    - One-way function
- If attacker gains database, practically impossible to take a hash value and directly determine the original password

Brute Force Attack on Hashed Passwords

- Aim: given one (or more) target hash value, find the original password
- Start with large set of possible passwords (e.g. from dictionary, all possible *n*-character combinations)
- Calculate hash of possible password, compare with target hash
  - if match, original password is found
  - else, try next possible password
- Attack duration depends on size of possible password set

Pre-calculated Hashes and Rainbow Tables

- How to speed up brute force attack? Use hash values calculated by someone else
- Possible passwords and corresponding hashes stored in database
- Attacker performs lookup on database for target hash
- How big is such a database of pre-calculated hashes?
  - In raw form, generally too big to be practical (100's, 1000's of TB)
  - Using specialised data structures (e.g. Rainbow tables), can obtain manageable size, e.g. 1 TB
- Trade-off: reduce search time, but increase storage space
- Countermeasures:
  - Longer passwords
  - Slower hash algorithms
  - Salting the password before hashing

Salting Passwords

$$ID, Salt, H(P||Salt)$$

- When ID and password initially created, generate random s-bit value (salt), concatenate with password and then hash
- When user submits password, salt from password database is concatenated, hashed and compared
- If attacker gains database, they know the salt; same effort to find password as brute force attack
- BUT pre-calculated values (e.g. Rainbow tables) are no longer feasible
  - Space required increased by factor of 2^s

Password Storage: Best Practice

When storing user login information, always store a hash of a salted password

$$ID, Salt, H(P||Salt)$$

- Password: see next sections on password policies
- Salt: random, generated when ID/password first stored; 32 bits or longer
- Hash function: slow, adaptive speed (work factor), e.g. bcrypt/scrypt, PBKDF2

Design for failure: assume password database will eventually be compromised

Contents

User Authentication

Password-Based Authentication

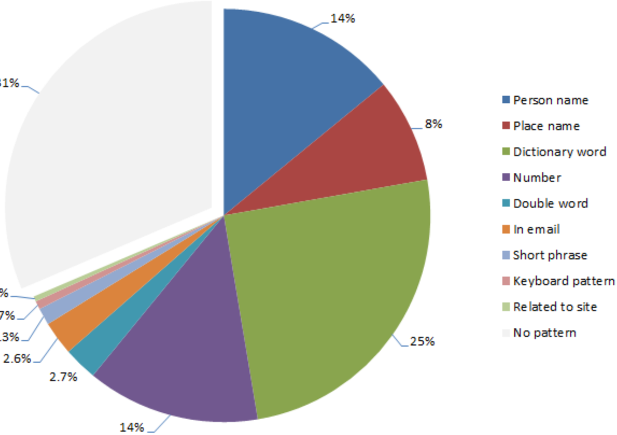
Password Entropy

Storing Passwords

Selecting Passwords

How Do People Select Passwords?

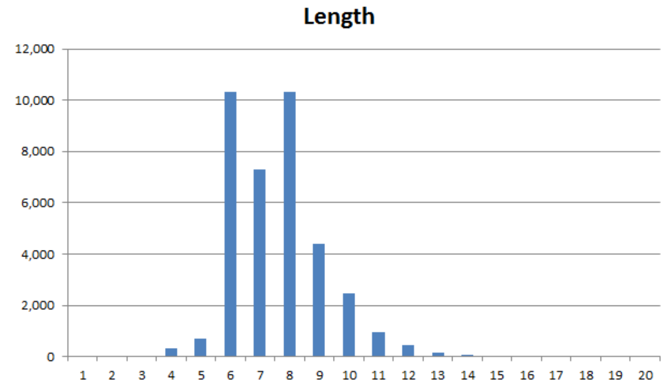
Analysis of 300,000 leaked passwords



Credit: Troy Hunt, *The science of password selection*, www.troyhunt.com, CCBY3.0

How Long Are Passwords?

Analysis of 37,000 leaked passwords



Credit: Troy Hunt, *A brief Sony password analysis*, www.troyhunt.com, CCBY3.0

Other Common Characteristics of Passwords

- Most use only alphanumeric characters
- Most are in (password) dictionaries
- Many users re-use passwords across systems
- Some very common passwords: 123456, password, 12345678, qwerty, abc123, letmein, iloveyou, . . .
- When forced to change passwords, most users change a single character

Password Selection Strategies

User education Ensure users are aware of importance of hard-to-guess passwords; advise users on strategies for selecting passwords

Computer-generated passwords Generate random or pronounceable passwords (but poorly accepted by users)

Reactive password checking Regularly check user's passwords, inform them if weak passwords

Proactive password checking Advise user on strength when selecting a password