CSE 410 Fall 2025
Privacy-Enhancing Technologies

Marina Blanton

Department of Computer Science and Engineering
University at Buffalo

Lecture 6: Protecting Data at Rest IV: Working with Passwords

Big Picture of Data Protection

The components we discussed:

- encryption (confidentiality protection)
- integrity protection
- key generation
- randomness generation

Big Picture of Data Protection

The components we discussed:

- encryption (confidentiality protection)
- integrity protection
- key generation
- randomness generation

The piece that remains:

- managing keys

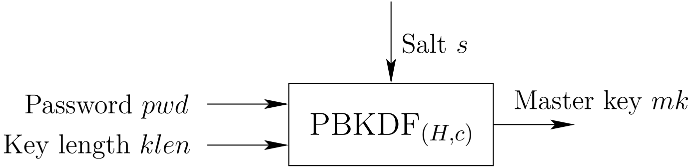
Password-Based Key Derivation

- In some applications, passwords may be the only required input for users eligible to access protected data
- Passwords have low entropy and poor randomness and are not suitable as cryptographic keys
- The solution is to use password-based key derivation
  - NIST recommendation is available as special publication SP 800-132

Key Derivation Function

- A key derivation function (KDF) derives key material from password, passphrase, key, etc.
  - it creates a master key mk, from which we derive data protection keys
  - password-based key derivation function (PBKDF) takes a string of characters (password or passphrase) chosen by a user
- With PBKDF, we also specify
  - the length of a key to produce $klen$ (≈ 128)
  - hash function H (e.g., SHA-256)
  - iteration count c (e.g., 100,000)

PBKDF



- the salt needs to be pseudo-random ($\geq$ 128 bits)
- the hash function H (to be used with HMAC) has to be strong
- let $hlen$ be the hash function output length

PBKDF Algorithm

- If $klen \leq hlen$, we compute

1. $t = 0$
2. $u_0 = s || 1$
3. for ($j = 1$ to c)
4. $\quad u_j = \text{HMAC}(p, u_{j-1})$
5. $\quad t = t \oplus u_j$
6. return $t[0 \ldots hlen - 1]$

PBKDF Algorithm

- Full algorithm:

1. $len = \lceil klen/hlen \rceil$
2. $r = klen - (len - 1) \cdot hlen$
3. for $(i = 1$ to $len)$
4. $\quad t_i = 0$
5. $\quad u_0 = s||i$
6. $\quad$ for $(j = 1$ to $c)$
7. $\quad\quad u_j = \text{HMAC}(p, u_{j-1})$
8. $\quad\quad t_i = t_i \oplus u_j$
9. return $mk = t_1||t_2||\ldots||t_{len}[0\ldots r-1]$

Password-Based Key Derivation

- One or more data protection keys (DPK) can be computed from the derived mk
  - a DPK can be a portion of the master key
  - a DPK can be computed from mk using a KDF
- A challenge is that the supplied password can be incorrect
  - there is a need to verify the validity of the generated DPK
  - option 1: add a fixed prefix to the data being decrypted
  - option 2: store the DPK protected (using authenticated encryption) with another key derived from mk

Password-Based Authentication

- Passwords are a simple mechanism for authenticating users
- A password serves the purpose of a shared secret between the user and the system
- A (userid, password) pair is associated with a user
  - userid identifies the user
  - password provides the necessary evidence that the user possesses the secret
- During authentication, the system compares user-supplied information with what it has stored
- Passwords are a weak form of authentication as they are prone to replay attacks

Password Storage

- Passwords are typically not stored in the clear
  - any data breach, insider attack makes all users vulnerable

Password Storage

- Passwords are typically not stored in the clear
  - any data breach, insider attack makes all users vulnerable
- Typically we hash a password using a one-way hash function and store the hash
  - stored information can look like

$$uid_1, H(pwd_1)$$
$$uid_2, H(pwd_2)$$

  - the password itself cannot be recovered, but there are other concerns

Attacks on (Hashed) Passwords

- Exhaustive search: an attacker attempts to guess a user password by trying all possible strings
  - this is most effective if the attacker has the password file
  - can be infeasible if the password space is large (but can exhaust all short passwords)
- Dictionary attack: an attacker tries to guess a password using words from a dictionary and variations thereof
  - can have a high probability of success
  - dictionary attacks are now sophisticated

Password Storage

One of the measures to decrease the vulnerability of the system is password salting

- this technique makes guessing attacks less effective
- a password is augmented with a random string, called salt, prior to hashing
- the salt is stored in cleartext in the password file

$$uid_1, salt_1, H(salt_1 || pwd_1)$$
$$uid_2, salt_2, H(salt_2 || pwd_2)$$

- how does it improve security?

Summary

- We extensively examined protecting data at rest
- Next, we turn to protecting data in transit