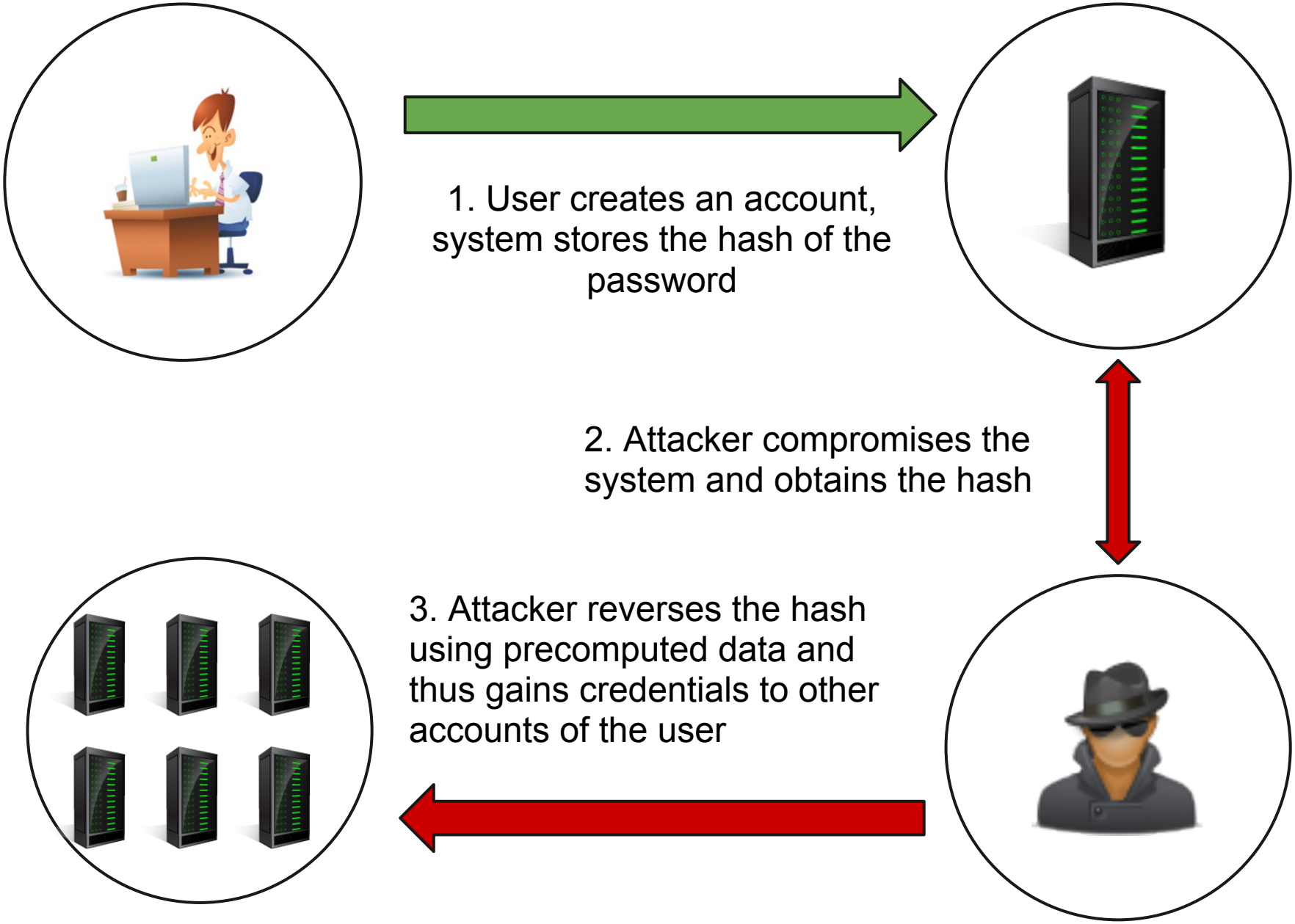
Ferhat Elmas
Abdulkadir Karaağaç
Javi Martin
Przemysław Pietrzkiewicz

One-way hashing without salt

"You're probably storing passwords incorrectly"

Storing user passwords

- Plaintext
- Hashed 🔒
- Hashed with salt 🔒🔒


1. User creates an account, system stores the hash of the password
2. Attacker compromises the system and obtains the hash
3. Attacker reverses the hash using precomputed data and thus gains credentials to other accounts of the user

Prerequsities:

- Weak user password (User)
- Hashing the passwords directly (System)
- Compromised system (System)
- Same password on a number of systems (User)

Effects:

- Attacker gains unauthorised access to other systems
  - User data, assets and reputation is threathened
  - Integrity and security of accessed systems is threatened

Vulnerable code:

```
register_user(login, password):
    hash = sha.new(password).hexdigest()
    save_hash(login, hash)

verify_credentials(login, password):
    hash = sha.new(password).hexdigest()
    if hash == load_hash(login):
        return Correct
    else
        return Incorrect
```

Better code:

```
register_user(login, password):
    salt = sha.new(str(random.random())).hexdigest()
    hash = sha.new(password+salt).hexdigest()
    save_hash(login, hash)
    save_salt(login, salt)

verify_credentials(login, password):
    salt = load_salt(login, salt)
    hash = sha.new(password+salt).hexdigest()
    if hash == load_hash(login):
        return Correct
    else
        return Incorrect
```

Defense measures:

- Salt the passwords before hashing
  - Preferably with different salt for each password
  - Preferably with decent-length salt
- Use specialised hashing algorithms that introduce work factor (such as **bcrypt**) instead of general ones (such as **sha** and **md5**) that are prone to bruteforce attack (with known hash **and salt**).

Q & A