Part II
Access Control

Access Control

- Two parts to access control
- **Authentication:** Who goes there?
  - Determine whether access is allowed
  - Authenticate human to machine
  - Authenticate machine to machine
- **Authorization:** Are you allowed to do that?
  - Once you have access, what can you do?
  - Enforces limits on actions
- Note: Access control often used as synonym for authorization

Chapter 7
Authentication

Who Goes There?

- How to authenticate a human to a machine?
- Can be based on…
  - Something you **know**
    - For example, a password
  - Something you **have**
    - For example, a smartcard
  - Something you **are**
    - For example, your fingerprint

Something You Know

- Passwords
- Lots of things act as passwords!
  - PIN
  - Social security number
  - Mother's maiden name
  - Date of birth
  - Name of your pet, etc.

Trouble with Passwords

- "Passwords are one of the biggest practical problems facing security engineers today."
- "Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They 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.)"

Why Passwords?

- Why is "something you know" more popular than "something you have" and "something you are"?
- **Cost**: passwords are free
- **Convenience**: easier for SA to reset pwd than to issue user a new thumb

Keys vs Passwords

- **Crypto keys**
- Spse key is 64 bits
- Then 2^{64} keys
- Choose key at random
- Then attacker must try about 2^{63} keys

- **Passwords**
- Spse passwords are 8 characters, and 256 different characters
- Then $256^8 = 2^{64}$ pwds
- Users do **not** select passwords at random
- Attacker has far less than 2^{63} pwds to try **(dictionary attack)**

Good and Bad Passwords

- Bad passwords
  - frank
  - Fido
  - password
  - 4444
  - Pikachu
  - 102560
  - AustinStamp
- Good Passwords?
  - jfIej,43j-EmmL+y
  - 09864376537263
  - P0kem0N
  - FSa7Yago
  - 0nceuP0nAt1m8
  - PokeGCTall150

Password Experiment

- Three groups of users —each group advised to select passwords as follows
  - **Group A:** At least 6 chars, 1 non-letter
  - **Group B:** Password based on passphrase
  - **Group C:** 8 random characters

winner →

- Results
  - **Group A:** About 30% of pwds easy to crack
  - **Group B:** About 10% cracked
    - Passwords easy to remember
  - **Group C:** About 10% cracked
    - Passwords hard to remember

Password Experiment

- User compliance hard to achieve
- In each case, 1/3rd did not comply (and about 1/3rd of those easy to crack!)
- Assigned passwords sometimes best
- If passwords not assigned, best advice is
  - Choose passwords based on passphrase
  - Use pwd cracking tool to test for weak pwds
  - Require periodic password changes?

Attacks on Passwords

- Attacker could…
  - Target one particular account
  - Target any account on system
  - Target any account on any system
  - Attempt denial of service (DoS) attack
- Common attack path
  - Outsider → normal user → administrator
  - May only require **one** weak password!

Password Retry

- Suppose system locks after 3 bad passwords. How long should it lock?
  - 5 seconds
  - 5 minutes
  - Until SA restores service
- What are +'s and -'s of each?

Password File

- Bad idea to store passwords in a file
- But need a way to verify passwords
- Cryptographic solution: **hash** the passwords
  - Store y = h(password)
  - Can verify entered password by hashing
  - If attacker obtains password file, he does not obtain passwords
  - But attacker with password file can guess x and check whether y = h(x)
  - If so, attacker has found password!

Dictionary Attack

- Attacker pre-computes h(x) for all x in a **dictionary** of common passwords
- Suppose attacker gets access to password file containing hashed passwords
  - Attacker only needs to compare hashes to his pre-computed dictionary
  - Same attack will work each time
- Can we prevent this attack? Or at least make attacker's job more difficult?

Password File

- Store hashed passwords
- Better to hash with **salt**
- Given password, choose random s, compute
  y = h(password, s)
  and store the pair (s,y) in the password file
- Note: The salt s is **not secret**
- Easy to verify password
- Attacker must recompute dictionary hashes for each user—lots more work!