# Password Vulnerability

# Access Control

- Two parts to access control…
- **Authentication:** Are you who you say you are?
  - Determine whether access is allowed or not
  - Authenticate human to machine
  - Or, possibly, 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

# Are You Who You Say You Are?

- 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.”

# Why Passwords?

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

# Keys vs Passwords

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

- **Passwords**
- Suppose 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
  - incorrect
  - 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
  - winner → **Group B:** Password based on passphrase
  - **Group C:** 8 random characters
- 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 $\rightarrow$ normal user $\rightarrow$ 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 we need to verify passwords
- Solution? **Hash** passwords
  - Store y = h(password)
  - Can verify entered password by hashing

- If Trudy obtains the password file, she does not (directly) obtain passwords
- But Trudy can try a *forward search*
  - Guess x and check whether y = h(x)

passw0rd

$\downarrow h_{MD5}$

BED128365216C019988915ED3ADD75FB

# Dictionary Attack

- Trudy pre-computes h(x) for all x in a **dictionary** of common passwords
- Suppose Trudy gets access to password file containing hashed passwords
  - She only needs to compare hashes to her pre-computed dictionary
  - After one-time work of computing hashes in dictionary, actual attack is trivial
- Can we prevent this forward search attack? Or at least make it more difficult?

# Salt


- Hash password with **salt**
- Choose random salt s and compute

$$y = h(\text{password}, s)$$

  and store (s,y) in the password file
- Note that the salt s is not secret
- Still easy to verify salted password
- But lots more work for Trudy
  - Why?

# Password Cracking: Do the Math

- Assumptions:
- Pwds are 8 chars, 128 choices per character
  - Then $128^8 = 2^{56}$ possible passwords
- There is a **password file** with $2^{10}$ pwds
- Attacker has **dictionary** of $2^{20}$ common pwds
- **Probability** 1/4 that password is in dictionary
- **Work** is measured by number of hashes

# Password Cracking: Case I

- Attack 1: specific password ***without*** using a dictionary
  - E.g., Alice's password
  - Must try $2^{56}/2 = 2^{55}$ on average
  - Like exhaustive key search
- Does **salt** help in this case?

# Password Cracking: Case II

- Attack 1 specific password ***with*** dictionary
- With **salt**
  - Expected work: $1/4\ (2^{19}) + 3/4\ (2^{55}) \approx 2^{54.6}$
  - In practice, try all pwds in dictionary…
  - …then work is at most $2^{20}$ and probability of success is 1/4
- What if **no salt** is used?
  - One-time work to compute dictionary: $2^{20}$
  - Expected work is of same order as above

# Password Cracking: Case III

- Attack3: Any of 1024 pwds in file, ***without*** dictionary
  - Assume all $2^{10}$ passwords are distinct
  - Need $2^{55}$ **comparisons** before expect to find pwd
- If **no salt** is used
  - Each computed hash yields $2^{10}$ comparisons
  - So expected work (hashes) is $2^{55}/2^{10} = 2^{45}$
- If **salt** is used
  - Expected work is $2^{55}$
  - Each comparison requires a hash computation

# Password Cracking: Case IV

- Attack 4: Any of 1024 pwds in file, ***with*** dictionary
  - Prob. one or more pwd in dict.: $1 - (3/4)^{1024} \approx 1$
  - So, we ignore case where no pwd is in dictionary
- What if **no salt** is used?
  - If dictionary hashes not precomputed, work is about $2^{19}/2^{10} = 2^9$
- If **salt** is used, expected work less than $2^{22}$
  - Work ≈ size of dictionary / P(pwd in dictionary)

$$\frac{1}{4}(2^{19}) + \frac{3}{4}.\frac{1}{4}(2^{20} + 2^{19}) + \left(\frac{3}{4}\right)^2 \frac{1}{4}(2.2^{20} + 2^{19}) + \cdots + \left(\frac{3}{4}\right)^{1023} \frac{1}{4}(1023.2^{20} + 2^{19})$$

$<2^{22}$

# Other Password Issues

- Too many passwords to remember
  - Results in password reuse
  - Why is this a problem?
- Who suffers from bad password?
  - Login password vs ATM PIN
- Failure to change default passwords
- Social Engineering
- Bugs, keystroke logging, spyware, etc.

# Passwords

- The bottom line…
- **Password attacks are too easy**
  - Often, one weak password will break security
  - Users choose bad passwords
  - Social engineering attacks, etc.
- Trudy has (almost) all of the advantages
- All of the math favors bad guys
- Passwords are a **BIG** security problem
  - And will continue to be a problem

# Password Cracking Tools

- Popular password cracking tools
  - Password Crackers
    - http://www.pwcrack.com/index.shtml
  - L0phtCrack and LC4 (Windows)
    - https://www.helpnetsecurity.com/2002/08/14/lc4/
    - https://www.l0phtcrack.com/doc/Introduction.html
  - John the Ripper (Unix)
    - http://www.openwall.com/john/
- System Admin should use these tools to test for weak passwords since attackers will
- Good articles on password cracking
  - Various password research articles are maintained in
    - http://passwordresearch.com/
  - Passwords revealed by sweet deal
    - http://news.bbc.co.uk/2/hi/technology/3639679.stm