Announcements

• HW 5 is out
  – Due: Friday, Nov. 12\textsuperscript{th}

• Midterm 2 will be held Monday, Nov. 15\textsuperscript{th}
Man vs. Machine

- **Machine**
  - Good at authenticating other machines
  - Good at mathematical manipulations, etc.
  - Can handle keys, secrets, etc.
  - Very good memory of things stored in it

- **Man**
  - Good at identifying people
  - Use small clues that when combined yield an unmistakable picture
    - Voice
    - Height
Authenticating Humans: Foundations

• Authentication is based on one or more of the following:
  • **Something you know**
    – password
  • **Something you have**
    – driver’s license, Penn Card
  • **Something inherent about you**
    – Biometrics, location

• What’s the most common method of authentication?
Passwords

- Shared code/phrase
- Client sends to authenticate

- Simple, right?
- How do you…
  - Establish them to begin with?
  - Stop them from leaking?
  - Stop them from being guessed?

http://www.captcha.net/
Prime Mover Problem

• Out of band
  – Physical mail
  – Email
  – Attached to the box

• Piggybacking
  – Swipe Penn Card to make PennKey
  – But where does the chain stop?
    • Penn Card -> drivers license -> birth certificate
Leaks

• Social engineering
  – Warnings

• Legal and responsibility
  – Shared password == shared liability

• Writing the password down on paper
Guessing

- The "no such user" mistake
- The "here's who we are" mistake
- Common words, phrases for passwords
- Null passwords, "password", username, backwards, etc.
- Dictionary attacks

- How bad is it?
1979 Survey of 3,289 Passwords

• With no constraints on choice of password, Morris and Thompson got the following results:
  – 15 were a single ASCII letter.
  – 72 were strings of two ASCII letters.
  – 464 were strings of three ASCII letters.
  – 47 were strings of four alphanumerics.
  – 706 were five letters, all upper-case or all lower-case.
  – 605 were six letters, all lower case.
1990s Surveys of 15K Passwords

• Klein (1990) and Spafford (1992)
  – 2.7% guessed in 15 minutes
  – 21% in a week
  – Sounds ok? Not if the passwords last 30 days

• Tricks
  – Letter substitutions, words backwards, common names, patterns, etc.
  – Anything you can think of off the top of your head, a hacker can think of too

• Lazy users!
  – Weakest link is always the way of the attack
Heuristics for Guessing Attacks

• The dictionary with the words spelled backwards

• A list of first names (best obtained from some mailing list). Last names, street names, and city names also work well.

• The above with initial upper-case letters.

• All valid license plate numbers in your state. (About 5 hours work in 1979 for New Jersey.)

• Room numbers, social security numbers, telephone numbers, and the like.
What makes a good password?

• Password Length
  – 64 bits of randomness is hard to crack
  – 64 bits is roughly 20 “common” ASCII characters
  – But… People can’t remember random strings
  – Longer not necessarily better: people write the passwords down

• Pass phrases
  – English Text has roughly 1.3 random bits/char.
  – Thus about 50 letters of English text
  – Hard to type without making mistakes!

• In practice
  – Non-dictionary, mixed case, mixed alphanumeric
  – Not too short (or too long)
Hacks on plaintext password file

• Is the password file readable by the OS?
  – Then if I break the OS

• Can privileged users see the file?
  – … and make copies

• Is the file backed up somewhere
  – … insecure?

• Is the file in plaintext somewhere in memory?
  – Core dump

• Fool the user
  – A program that masquerades as the authentication program
Counter-hacks

• Control-Alt-Del for logging in
  – For windows only

• Slow down
  – Make guessing take too long

• Encrypt the password file
  – “Salt" - to prevent duplicates
  – Use one way hashes or encryptions on the passwords

• Password rules
  – Min length, upper and lower case, no common words
  – Use letters and numbers and symbols
  – Change often
  – Keep a password history
  – Don’t write it down!
Add Salt

• “Salt” the passwords by adding random bits.
  – Decreases the likelihood that two identical passwords will appear as identical entries in the password file.
• 12 bit salt results in 4,096 versions of each password.
• /etc/passwd entry:

  
  
  user_id | salt_u | Hash(salt_u + passwd_u) | ...

  
  

• Actually most modern implementations use so-called shadow password files /etc/shadow that aren’t world readable.
Where does this lead us?

• Turn humans into machines?
  – We want long, random, non-language passwords for security
  – But we can't remember them

• Machine aided password generation
  – Vax password suggestions
  – One time passwords
One Time Passwords

• Shared lists.
• Sequentially updated.
• One-time password sequences based on a one-way (hash) function.

• Used in practice: SKey mechanism
Hash-based 1-time Passwords

- Alice identifies herself to verifier Bart using a well-known one-way hash function $H$.

- One-time setup.
  - Alice chooses a secret $w$.
  - Fixes a constant $t$ for the number of times the authentication can be done.
  - Alice securely transfers $H^t(w)$ to Bart
    \[
    H(H(H(...(H(w))...)))
    \]
    $t$ times
Hash-based 1-time Passwords

• Protocol actions. For session i, claimant A does the following to identify itself:
  – A computes \( w' = H^{(t-i)}(w) \) and transmits the value to B.
  – B checks that i is the correct session (i.e. that the previous session was i-1) and checks to see if \( H(v) = w' \) where v was the last value provided by A (as part of session i-1).
  – B saves \( w' \) and i for use in the next session.
One-time passwords: $i^{th}$ authentication

Alice does the following to identify herself:
- $A$ computes $w' = H^{(t-i)}(w)$ and transmits the value to $B$.
- $B$ checks that $i$ is the correct session (i.e., that the previous session was $i-1$) and checks to see if $H(w') = v$ where $v$ was the last value provided by $A$ (as part of session $i-1$).
- $B$ saves $w'$ and $i$ for use in the next session.
Why This 1-time Password Works

• It’s hard to compute $x$ from $H(x)$.
  – Even though attacker gets to see $H^{(t-i)}(x)$, they can’t guess then next message $H^{(t-(i+1))}(x)$. 