Announcements

- Reading Chapter 19

- Midterm Results

<table>
<thead>
<tr>
<th></th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>12</td>
<td>10</td>
<td>67.2</td>
</tr>
<tr>
<td>Min</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>Max</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>20</td>
<td>15</td>
<td>87</td>
</tr>
<tr>
<td>stddev</td>
<td>4.75</td>
<td>5.6</td>
<td>5.1</td>
<td>5.3</td>
<td>4.2</td>
<td>11</td>
</tr>
</tbody>
</table>
Authentication

- How does the computer know who is using it?
  - need to exchange some information to verify the user
  - types of information exchanged:
    - pins
      - numeric passwords
      - too short to be secure in most cases
    - passwords
      - a string of letters and numbers
      - often easy to guess
    - challenge/response pairs
      - user needs to be apply to apply a specific algorithm
      - often involve use of a calculator like device
      - can be combined with passwords
    - unique attributes of the person
      - i.e. signature, thumb print, DNA?
      - sometimes these features can change during life
Authentication (cont.)

- How does a user know what computer they are using?
- Need to have *mutual authentication*
  - computer presents some information that only it could contain
  - example: NT <ctrl>-<alt>-<del> to login
    - user software can’t trap that information
    - assumes that the kernel itself is secure
- telephone example:
  - never give banking/credit card info over the phone unless you placed the phone call
    - i.e. you use the telco namespace for authentication
Example (UNIX passwords)

- **use a function that is hard to invert**
  - “easy” to compute $f(x)$ given $x$
  - hard to compute $x$ given $f(x)$
  - the function used is a variation on the DES algorithm
    - changes selected items in the transformation matrix to prevent hardware attacks
  - store only $f(x)$ in the filesystem

- **to login:**
  - user supplies a password $x'$
  - compute $f(x')$ and compare to $f(x)$

- **salt**
  - add an extra two characters to $x$ so that the same $x$ will produce different values on different machines

- **dictionary attack**
  - if it's too easy to compute $f(x)$
  - can “guess” many passwords and try them out
Types of Software Threats

● **Trojan Horse**
  - a program that looks like a normal program
  - for example a login program written by a user
  - UNIX example: never put “.” early in your path

● **Trap door**
  - hole left by the programmers to let them into the system
  - “system” password set to a default value by the vendor

● **Worms**
  - programs that clone themselves and use resources
  - Internet worm:
    - exploited several bugs and “features” in UNIX
      - .rhosts files
      - bug in finger command (overwrite strings)
      - sendmail “debug” mode to run commands