Announcements

- Reading Chapter 19
- MT#2 re-grade requests due by Tuesday
  - Must be submitted in writing to grades.cs.umd.edu
  - Provide paper copy of exam to me
### Summary of Scores to Date

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>MT1</th>
<th>P3</th>
<th>P4</th>
<th>MT2</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>50</td>
<td>52</td>
<td>51</td>
<td>52</td>
<td>51</td>
<td>49</td>
<td>49</td>
<td>52</td>
</tr>
<tr>
<td><strong>Minimum</strong></td>
<td>90</td>
<td>30</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td><strong>Maximum</strong></td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>93</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>99.4</td>
<td>96.15</td>
<td>87.25</td>
<td>63.19</td>
<td>65.2</td>
<td>51.84</td>
<td>67.41</td>
<td>35.93</td>
</tr>
<tr>
<td><strong>Std Dev</strong></td>
<td>2.18</td>
<td>14.16</td>
<td>22.37</td>
<td>13.88</td>
<td>30.11</td>
<td>30.8</td>
<td>18.32</td>
<td>8.55</td>
</tr>
</tbody>
</table>
Computer Threat Model

- must consider acceptable risks
  - value of item to be protected
  - $2,000 of computer time to steal 50 cents of data
    - this is a sufficient deter someone
    - **but** computers keep getting faster
- Basic Ideas:
  - confine access to only the highest level needed
    - run programs as root only if needed
    - don’t give system access to all users
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: Windows <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 attach
  - 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
Viruses

- **Most common on systems with little security**
  - easy to write to boot blocks, system software
  - never run untrusted software with special privileges
  - Don’t perform daily operations with root/system privileges

- **Possible to write system independent viruses**
  - MS Word virus
    - uses macros to call into the OS
Access Matrix

● Abstraction of protection for objects in a system.
  – Rows are domains (users or groups of users)
  – Columns are objects (files, printers, etc.)
  – Items are methods permitted by a domain on an objects
    • read, write, execute, print, delete, …

● Representing the Table
  – simple representation (dense matrix) is large
  – sparse representation possible: each non-zero in the matrix
  – observation: same column used frequently
    • represent groups of users with a name and just store that
  – create a default policy for some objects without a value

● Revocation of access
  – when are access rights checked?
  – selective revocation vs. global
## Access Matrix

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>Laser Printer</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>read</td>
<td></td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td>execute</td>
<td>print</td>
</tr>
<tr>
<td>D3</td>
<td>read, write</td>
<td></td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td></td>
<td></td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td></td>
<td>delete</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Rows represent users or groups of users
- Columns represent files, printers, etc.
Capabilities

- Un-forgeable Key to access something
- Implementation: a string
  - I.e. a long numeric sequence for a copier
- Implementation: A protected memory region
  - tag memory (or procedures) with access rights
    - example - x86 call gate abstraction
  - permit rights amplification