Announcements

- Reading Chapter 18 (Security)
- Midterm #2 was returned
- Project #6 is on the web
  - It is a continuation of #5 and requires your #5 to work
Midterm #2

- Min: 21, Max: 91, Mean: 59, Std. Dev: 17
- Per question breakdown:

<table>
<thead>
<tr>
<th>Question</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>20</td>
<td>9.4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>20</td>
<td>9.3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>20</td>
<td>12.3</td>
</tr>
</tbody>
</table>
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 or MD5 algorithms
    - 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
  - need to store salt along with password

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

- **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
  – HTML (javascript)
  – Flash
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 object
    - 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></td>
<td>delete</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