Advanced Systems Security: Ordinary Operating Systems

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

February 2, 2010
UNIX and Windows

- If you want to run an application, you have to install one of these
  - Where UNIX encompasses Linux and Mac OS X
- Common understanding
  - They are insecure
- Why?
UNIX Access Control

• On Files
  ‣ All objects are files
  ‣ Not exactly true

• Classical Protection System
  ‣ Limited access matrix
  ‣ Discretionary protection state operations

• Practical model for end users
  ‣ Still involves some policy specification
## UNIX Mode Bits

<table>
<thead>
<tr>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-rw-r--</td>
<td>1 pbg</td>
<td>staff</td>
<td>31200</td>
<td>Sep 3</td>
<td>08:30</td>
<td>intro.ps</td>
</tr>
<tr>
<td>drwx---------</td>
<td>5 pbg</td>
<td>staff</td>
<td>512</td>
<td>Jul 8</td>
<td>09:33</td>
<td>private/</td>
</tr>
<tr>
<td>drwxrwxr-x</td>
<td>2 pbg</td>
<td>staff</td>
<td>512</td>
<td>Jul 8</td>
<td>09:35</td>
<td>doc/</td>
</tr>
<tr>
<td>drwxrwx---</td>
<td>2 pbg</td>
<td>student</td>
<td>512</td>
<td>Aug 3</td>
<td>14:13</td>
<td>student-proj/</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>1 pbg</td>
<td>staff</td>
<td>9423</td>
<td>Feb 24</td>
<td>2003</td>
<td>program.c</td>
</tr>
<tr>
<td>-rw-r-xr-x</td>
<td>1 pbg</td>
<td>staff</td>
<td>20471</td>
<td>Feb 24</td>
<td>2003</td>
<td>program</td>
</tr>
<tr>
<td>drwx--x--x</td>
<td>4 pbg</td>
<td>faculty</td>
<td>512</td>
<td>Jul 31</td>
<td>10:31</td>
<td>lib/</td>
</tr>
<tr>
<td>drwx---------</td>
<td>3 pbg</td>
<td>staff</td>
<td>1024</td>
<td>Aug 29</td>
<td>06:52</td>
<td>mail/</td>
</tr>
<tr>
<td>drwxrwxrwx</td>
<td>3 pbg</td>
<td>staff</td>
<td>512</td>
<td>Jul 8</td>
<td>09:35</td>
<td>test/</td>
</tr>
</tbody>
</table>
Windows Access Control

- **On Objects**
  - Arbitrary classes can be defined
  - New classes can be defined (Active Directory)

- **Classical Protection System**
  - Full-blown ACLs (even negative ACLs)
  - *Discretionary* protection state operations

- **Not so usable**
  - Few people have experience
Windows Access Control

Diagram showing the relationship between Process or Thread, Access Token, Security Descriptor, and DACL. Access is denied for a user with a SID, while access is allowed for another user with a different SID.
Vulnerabilities

• Function run by one subject that enables an attacker to gain unauthorized privileges (i.e., the subject’s)
  ‣ Privilege escalation

• Two views
  ‣ Subject could do something for the attacker
  ‣ Attacker could “take over” process

• Either way, the attacker can use the subject’s privileges
  ‣ May use these to compromise an even more privileged subject
The Morris Worm

• Robert Morris, a 23 year old doctoral student from Cornell
  – Wrote a small (99 line) program
  – November 3rd, 1988
  – Simply disabled the Internet

• How it did it -- summary
  – Buffer overflow on the fingerd program -- has root access
  – Cracks user passwords on the compromised host to try on other hosts
  – Used liberal system configurations of the time to identify hosts that accept logins without a password from the compromised host
  – Scanned local interfaces for network information
  – Covered its tracks (set is own process name to sh, prevented accurate cores, re-forked itself)

• Became significant because it would re-attack an already infected system after 7 scans
Confused Deputy

• A server process has the privileges necessary to service requests from all its clients
  ‣ Can a client trick the server into using its permissions for the client?

• Have the server access an object whose name is supplied by a client
Links

• Multiple names for a single inode
• Run
  – ln -s /etc/passwd badlink
  – setuid_program badlink < *passwd-entry*
  – To overwrite /etc/passwd
• Programs have to be aware of which files they are using
• open(file, O_NOFOLLOW, mode)
  – Prevents open from following a link
Unknown Origin

• A server may use a file under the control of a client
  ‣ Client may provide file by name or predict the name of a file the server may create/use

• Such a file provides guidance that the server trusts
  ‣ Enabling the client to control how the server runs
/tmp Vulnerability

• creat(pathname, mode)
• O_EXCL flag
  – if file already exists this is an error
• Potential attack
  – Create file in shared space (/tmp)
  – Give it a filename used by a higher authority service
  – Make sure that service has permission to the file
  – If creat is used without O_EXCL, then can share the file with the higher authority process
Others

- Overflow (Buffer or Heap)
  - Inject code into server
- Libraries (Path or input)
  - Inject code into server
- TOCTTOU
  - Cause an authorization to pass
- Integer overflow
  - Cause different control path
The OS Will Protect Me

• User-space vulnerabilities are expected
  ‣ Those processes are untrusted

• OS policies will protect the system from harm
  ‣ Only the OS and a few processes need to be trusted

• Just need to specify the access control policy
  ‣ We can specify anything we want
  ‣ We have an access matrix
Secrecy

• Does the following protection state ensure the secrecy of J’s private key in $O_1$?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J$</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>$S_2$</td>
<td>N</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>$S_3$</td>
<td>N</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>
Integrity

- Does the following access matrix protect the integrity of J’s public key file $O_2$?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>$S_2$</td>
<td>N</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>N</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
Trusted vs. Benign

- Does it matter if we do not trust some of J’s processes?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>$N$</td>
<td>$R$</td>
<td>$R$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$N$</td>
<td>$R$</td>
<td>$R$</td>
</tr>
</tbody>
</table>
Protection and Security

- Protection
  - Security goals met under *benign* processes
  - Protects against error by non-malicious entity

- Security
  - Security goals met under *trusted* processes
  - Any benign process can come under the control of an attacker
  - Protects against any malicious entity

- For J: A benign process won’t accidentally leak a key, but it would under attackers’ control
Is Fixing the Policy Enough?

- No, as these systems do not satisfy the reference monitor concept
Complete Mediation

- **Mediation**: Does interface mediate correctly?
  - No. Several operations impact security that are ignored.

- **Mediation**: On all resources?
  - UNIX: No. No network.
  - Windows: Could.

- **Mediation**: Verifably?
  - Ha.
**Tamperproof**

- **Tamperproof**: Is reference monitor protected?
  - Operating system is not protect (see Rootkits)
  - Kernel modules, trusted processes, extensible function
  - Policy can be modified by untrusted processes (discretionary)

- **Tamperproof**: Is system TCB protected?
  - We don’t really know what this is
  - All root and setuid processes and ones they depend on
  - Plus anything an admin runs as root
Verification

- **Verifiable**: Is TCB code base correct?
  - No.

- **Verifiable**: Does the protection system enforce the system’s security goals?
  - Goals?
  - See Protection v. Security again
Take Away

• Conventional operating systems are insecure

• They run programs that suffer from many types of vulnerabilities

• They are designed to enable protection under benign programs, not secure a system from a directed attacker

• They do not satisfy reference monitor concept
  ‣ Also fail to implement a mandatory protection system