Privilege Separation

Dawn Song
dawnsong@cs.berkeley.edu

Part III OS Security

- **Principles of OS Security**
- **State-of-the-art techniques & tools for OS Security**

The Story Continues…

- **Programs have vulnerabilities and may not find all of them ahead of time**
- **What can we do?**
  - **Build security mechanisms to minimize damage**
- **Examples**
  - **Priviledge separation to prevent priviledge escalation**
  - **Isolation to protect other parts of the program and other programs**
  - **Sandboxing to limit the damage it does to the system**
  - **General concept: reference monitor**

Operating System & Privilege

- **OS's role**
  - Interface between hardware & applications
  - Manages resources
  - Provide protection to hardware & applications
- **Privilege**
  - Rights to perform certain operations
    - » E.g., writes to certain files & certain network operations

Principle of Least Privilege

- **Give the user/program only the privilege it needs to get its task done**
  - One of the most important principles in systems security
- **Why?**
  - Limit the damage when program misbehaves or is compromised
- **What privileges should you give to your**
  - ssh server
  - Video game program

Mangement of Privileges

- **Example: File privileges**
- **Superuser/root mode**
  - Access to everything
- **Windows privilege model**
  - Previously, most programs require superuser mode to install/run
    - » Consequence: most users log on as administrator
  - Vista: User Account Control (UAC)
    - » When user log on as a standard user, a token with basic privileges is assigned
    - » When user log on as an administrator, two tokens are assigned
      - One with basic privileges, the other with root privileges
      - Normal programs will be started with basic privileges
      - Programs require root privileges will be prompted for user permission

Privileged Programs

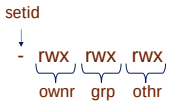
- Privilege management is coarse-grained in today's OS
  - Root can do anything
- Many programs run as root
  - Even though they only need to perform a small number of priviledged operations
- What's the problem?
  - Privileged programs are juicy targets for attackers
  - By finding a bug in parts of the program that do not need privilege, attacker can gain root

What Can We Do?

- Drop privilege as soon as possible
- Ex: a network daemon only needs privilege to bind to low port # (<1024) at the beginning
  - Solution?
  - Drop privilege right after binding the port
- What benefit do we gain?
  - Even if attacker finds a bug in later part of the code, can't gain privilege any more
- How to drop privilege?
  - Setuid programming in UNIX

Unix file permission

- Each file has owner and group
- Permissions set by owner
  - Read, write, execute
  - Owner, group, other
  - Represented by vector of four octal values



- Only owner, root can change permissions
  - This privilege cannot be delegated or shared
- Setid bits
  - Talk about this in a sec

Effective user id (EUID) in UNIX

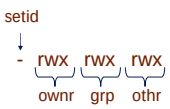
- **Each process has three Ids**
  - **Real user ID (RUID)**
    - **same as the user ID of parent (unless changed)**
    - **used to determine which user started the process**
  - **Effective user ID (EUID)**
    - **from set user ID bit on the file being executed, or sys call**
    - **determines the permissions for process**
      - **file access and port binding**
  - **Saved user ID (SUID)**
    - **So previous EUID can be restored**
- **Real group ID, effective group ID, used similarly**

Operations on UIDs

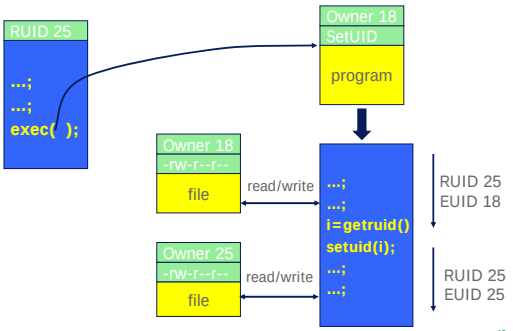
- **Root**
  - **ID=0 for superuser root; can access any file**
- **Fork and Exec**
  - **Inherit three IDs, except exec of file with setuid bit**
- **Setuid system calls**
  - **seteuid(newid) can set EUID to**
    - **Real ID or saved ID, regardless of current EUID**
    - **Any ID, if EUID=0**
- **Details are actually more complicated**
  - **Several different calls: setuid, seteuid, setreuid**

Setid bits on executable Unix file

- **Three setid bits**
  - **Setuid – set EUID of process to ID of file owner**
  - **Setgid – set EGID of process to GID of file**
  - **Sticky**
    - **Off: if user has write permission on directory, can rename or remove files, even if not owner**
    - **On: only file owner, directory owner, and root can rename or remove file in the directory**



Drop Privilege



Administravia

- **Photo submission due Oct 29**
- **Project 2**

What Happens if you can't drop privilege?

- **In what example scenarios does this happen?**
  - **A service loop**
  - **E.g., ssh**
- **Solution?**
  - **Privilege separation**
  - **Identifying operations that need privileges**
  - **Separate original code into master (priviledged) and slave (unprivileged)**
- **Example: ssh**

Privilege Separation

- **Process:**
  - Step 1: Identify which operations require privilege
  - Step 2: rewrite programs into 2 or more parts
- **Approach:**
  - Manual
    - » Have been done on security-critical programs, e.g., ssh
    - » Labor-intensive and may miss privileged operations
  - Automatic
    - » Automatic inference of privileged operations using a few initial annotations
    - » Automatic source-to-source rewriting
      - Privileged code move into master
      - Unprivileged code move into slave
      - Stubs for inter communication

16

Automatic Privilege Separation

- **Step 1: automatic inference of privileged data and operations**
  - Given some initial annotations of privileged data and/or operations, infer what other data/operations are privileged
  - Idea: can be viewed as a form of static taint analysis
  - Approach:
    - » Define qualifier _priv_ and _unpriv_
    - » Operations on _priv_ results in _priv_

```
int _priv_ a;
Int _priv_  f();
int b = f(a);          _priv_ b
c= c+b;                _priv_ c
g(c);                  _priv_ g
```

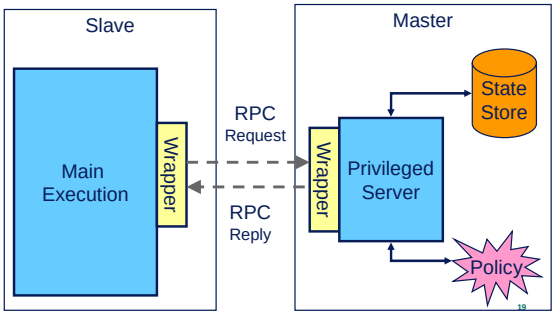
17

Automatic Privilege Separation

- **Step 2: automatic source-to-source transformation**
  - Move privileged data and code to Master
  - For call to privileged functions, change the call site to a wrapper function which marshals the args on slave side and sends them to Master's stub
  - Similar stubs on returns for unprivileged return values

18

Privilege Separation at Runtime



Summary: Privilege Separation

- **Only master is privileged, usually much smaller**
- **Slave is unprivileged**
- **Bug in slave cannot harm master, cannot gain privilege**
- **How to protect master from a compromised slave?**
  - **Fault isolation: e.g., running in different processes**