Operating Systems

Christopher Kruegel
Department of Computer Science
UC Santa Barbara
http://www.cs.ucsb.edu/~chris/
Operating Systems Security
Why do we care about operating systems (OS) security

- protect different applications that run at the same time
- applications may belong to different users, have different privileges
- keep buggy/malicious apps. from crashing each other
- keep buggy/malicious apps. from tampering with each other
- keep buggy/malicious apps. from crashing the OS

OS provides security services

- isolation (between processes)
- access control (regulates who can access which resources)
Operating Systems

- **Kernel**
  - provides an hardware abstraction layer for user-space programs
  - complete access to all (physical) resources
  - trusted computing base

- **Dual mode operation**
  - hardware (processor) support
  - when in kernel-mode, can do anything (direct hardware access)
  - when in user-mode, restricted access
  - typically, mode of operation is indicated by processor status bit(s)
  - of course, this bit can only be directly manipulated in kernel-mode
Operating Systems

Transition between different modes
– this crosses the border between two security domains
– clearly, a security relevant action

• System calls
– performs a transition from user mode to privileged (kernel) mode
– usually implemented with hardware (processor) support
  • processor interrupt (int 0x80)
  • x86 call gates (far call)
  • fast system call features (sysenter)
– ensure that only specific kernel code can be invoked
  • why not allow arbitrary calls into kernel code?
• Memory protection
  – through virtual memory abstraction
  – every process gets its own virtual memory space
  – no direct access to physical memory
  – page tables and memory MMU perform translation

• Programs are isolated and cannot talk to each other directly

• Inter-process communication
  – in some cases, shared memory can be requested
  – pipes, messages (packets) -> input validation necessary
  – file system (which is shared state) -> race conditions
• Other type of memory protection
  – physical memory can also be accessed via DMA (devices attached to bus)
  – several attacks have been published based on this
    • attack of the iPods
  – idea of I/O MMU comes to rescue
Operating Systems

• Access control
  – determine the actions that a process (subject) may perform on resources (objects)
  – requires to establish “identity” of subjects
  – implemented as access control lists (ACL) on objects; or capabilities carried by subjects

• Establishing identity
  – process of authentication
  – via something that one has, that one knows, or that one is (does)
  – should be protected by a trusted path
Operating Systems

• Discretionary access control
  – common model for contemporary operating systems
  – subject (owner) can change permission of objects

• Mandatory access control
  – less common, but gains popularity
  – enforced by the OS when subject cannot change permissions of objects
  – often associated with multi-level security (MLS) systems and the Bell-LaPadula model
Unix (Posix) Security
Unix

- Kernel vulnerability
  - usually leads to complete system compromise
  - attacks performed via system calls

- Solaris / NetBSD call gate creation input validation problem
  - malicious input when creating a LDT (x86 local descriptor table)
  - used in 2001 by Last Stage of Delirium to win Argus Pitbull Competition

- Kernel Integer Overflows
  - FreeBSD procfs code (September 2003)
  - Linux brk() used to compromise debian.org (December 2003)
  - Linux setsockopt() (May 2004)

- Linux Memory Management
  - mremap() and munmap() (March 2004)
• More recent Linux vulnerabilities
  – Linux message interface (August 2005, CAN-2005-2490)
  – race condition - proc and prctl (July 2006, CVE-2006-3626)
  – local privilege escalation - (September 2007, CVE 2007-4573)

• Device driver code is particularly vulnerable
  – (most) drivers run in kernel mode, either kernel modules or compiled-in
  – often not well audited
  – very large code based compared to core services

• Examples
  – aironet, asus_acpi, decnet, mpu401, msnd, and pss (2004)
    found by sparse (tool developed by Linus Torvalds)
  – remote root (MadWifi - 2006, Broadcom - 2006)
Unix

• Code running in user mode is **always** linked to a certain identity
  – security checks and access control decisions are based on user identity

• Unix is user-centric
  – no roles

• User
  – identified by user name (UID), group name (GID)
  – authenticated by password (stored encrypted)

• User **root**
  – superuser, system administrator
  – special privileges (access resources, modify OS)
  – cannot decrypt user passwords
Process Management

• Process Attributes
  – process ID (PID)
    • uniquely identified process
  – user ID (UID)
    • ID of owner of process
  – effective user ID (EUID)
    • ID used for permission checks (e.g., to access resources)
  – saved user ID (SUID)
    • to temporarily drop and restore privileges
  – lots of management information
    • scheduling
    • memory management, resource management
User Authentication

• How does a process get a user ID?
  ➢ Authentication (login)

• Passwords
  – user passwords are used as keys for crypt() function
  – runs DES algorithm 25 times on a block of zeros
  – 12-bit “salt”
    • 4096 variations
    • chosen from date, not secret
    • prevent same passwords to map onto same string
    • make dictionary attacks more difficult

• Password cracking
  – dictionary attacks
  – Crack, JohnTheRipper
User Authentication

• **Shadow passwords**
  – password file is needed by many applications to map user ID to user names
  – encrypted passwords are not

• `/etc/shadow`
  – holds encrypted passwords
  – account information
    • last change date
    • expiration (warning, disabled)
    • minimum change frequency
  – readable only by superuser and privileged programs
  – MD5 hashed passwords (default) to slow down guessing
File System

• File tree
  – primary repository of information
  – hierarchical set of directories
  – directories contain file system objects (FSO)
  – root is denoted “/”

• File system object
  – files, directories, symbolic links, sockets, device files
  – referenced by *inode* (index node)
File System

- **Access Control**
  - permission bits
  - `chmod`, `chown`, `chgrp`, `umask`
  - file listing:
    
    - `rwx  rwx  rwx`
    
    (file type) (user) (group) (other)

<table>
<thead>
<tr>
<th>Type</th>
<th>r</th>
<th>w</th>
<th>x</th>
<th>s</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>read access</td>
<td>write access</td>
<td>execute</td>
<td>suid / sgid</td>
<td>sticky bit</td>
</tr>
<tr>
<td>Directory</td>
<td>list files</td>
<td>insert and remove files</td>
<td>stat / execute files, chdir</td>
<td>new files have dir-gid</td>
<td>files only deleteable by owner</td>
</tr>
</tbody>
</table>
SUID Programs

- Each process has real and effective user/group ID
  - usually identical
  - real IDs
    - determined by current user
    - login, su
  - effective IDs
    - determine the "rights" of a process
    - system calls (e.g., setuid())
  - suid/sgid bits
    - to start process with effective ID different from real ID
    - attractive target for attacker

- Never use SUID shell scripts (multiplying problems)
Shell

- Shell
  - one of the core Unix application
  - both a command language and programming language
  - provides an interface to the Unix operating system
  - rich features such as control-flow primitives, parameter passing, variables, and string substitution
  - communication between shell and spawned programs via redirection and pipes
  - different flavors
    - bash and sh, tcsh and csh, ksh
Shell Attacks

- Environment Variables
  - $HOME and $PATH can modify behavior of programs that operate with relative path names
  - $IFS – internal field separator
    - used to parse tokens
    - usually set to \[ \t\n\] but can be changed to "/"
    - "/bin/ls" is parsed as "bin ls" calling bin locally
    - IFS now only used to split expanded variables
  - preserve attack (/usr/lib/preserve is SUID)
    - called "/bin/mail" when vi crashes to preserve file
    - change IFS, create bin as link to /bin/sh, kill vi
Shell Attacks

- Control and escape characters
  - can be injected into command string
  - modify or extend shell behavior
  - user input used for shell commands has to be rigorously sanitized
  - easy to make mistakes
  - classic examples are `;` and `&`

- Applications that are invoked via shell can be targets as well
  - increased vulnerability surface

- Restricted shell
  - invoked with `-r`
  - more controlled environment
Shell Attacks

- **system(char *cmd)**
  - function called by programs to execute other commands
  - invokes shell
  - executes string argument by calling `/bin/sh -c string`
  - makes binary program vulnerable to shell attacks
  - especially when user input is utilized

- **popen(char *cmd, char *type)**
  - forks a process, opens a pipe and invokes shell for cmd
File Descriptor Attacks

• SUID program opens file

• forks external process
  – sometimes under user control

• on-execute flag
  – if close-on-exec flag is not set, then
    new process inherits file descriptor
  – malicious attacker might exploit such weakness

• Linux Perl 5.6.0
  – getpwuid() leaves /etc/shadow opened (June 2002)
  – problem for Apache with mod_perl
Resource Limits

- File system limits
  - *quotas*
  - restrict number of storage blocks and number of inodes
  - hard limit
    - can never be exceeded (operation fails)
  - soft limit
    - can be exceeded temporarily
  - can be defined per mount-point
  - defend against resource exhaustion (denial of service)

- Process resource limits
  - number of child processes, open file descriptors
Signals

• **Signal**
  – simple form of interrupt
  – asynchronous notification
  – can happen anywhere for process in user space
  – used to deliver segmentation faults, reload commands, ...
  – `kill` command

• **Signal handling**
  – process can install signal handlers
  – when no handler is present, default behavior is used
    • ignore or kill process
  – possible to catch all signals except SIGKILL (-9)
Signals

• Security issues
  – code has to be re-entrant
    • atomic modifications
    • no global data structures
  – race conditions
  – unsafe library calls, system calls
  – examples

• Secure signals
  – write handler as simple as possible
  – block signals in handler
Windows Security
Windows

• > 90% of all computers run Windows
  – when dealing with security issues, it is important to have (some) knowledge of Windows
  – good example of non-open source system and security issues

• Started in 1985
  – graphical add-on to MS DOS

• Two main families
  – building on DOS legacy
    Windows 1.0, Windows 3.11, Windows 95, Windows ME
  – NT line (true 32 bit, multi-user OS)
    started with NT 3.1, NT 4.0, Windows 2K, XP, Vista
Windows NT

- Competitor to Unix
  - true multi-user
  - emphasis on portability and object-oriented design
  - isolation for applications and resource access control
  - similar to Unix, kernel and user mode

<table>
<thead>
<tr>
<th>User mode</th>
<th>System support processes (daemons)</th>
<th>Environment subsystems (csrss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System DLLs (ntdll, user32, kernel32, gdi32)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel mode</th>
<th>Executive (system call handlers, mem, procs, I/O, security monitor)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NT (Micro)-Kernel</td>
</tr>
<tr>
<td></td>
<td>Hardware and Hardware Abstraction Layer (HAL)</td>
</tr>
</tbody>
</table>
Windows NT

Important system processes

- Session Manager (similar to init)
- Client Server Runtime Process (Win32)
- Windows Logon Process (login)
- Service Control Manager (SCM)
- Local security authentication (LSA) process
Windows NT

Security Components

• Security Reference Monitor (SRM)
  – kernel process
  – performs access control decisions
  – generates security context

• Local Security Authentication (LSA)
  – user process
  – manages security policies (permission settings)
  – user authentication

• Windows Logon
  – user process
  – gather login information
Access Control Decisions

- **Object**
  - Windows is object-oriented, everything is an object
  - each object has security settings (*security descriptor*)

- **Subject**
  - threads / processes
  - have a *security context*

- **Operation**
  - determines desired access (read, write, delete, …)

- **Access Control Decision**
  - determines whether object permits certain operations for security context
  - implemented by SRM functionality (SeAccessCheck)
  - if access is permitted, typically an object handle is returned
Security Context

• Security Context
  – stored in (access) token
  – associated with every thread / process

• Access token
  – kernel data structure that determines rights of a subject
  – important fields
    • User SID (Security IDentifiers)
    • Group SIDs
    • Privileges
    • Default permissions (used for files that are created)
    • Management information
Security Identifiers (SID)

• Secure Identifiers
  – used to uniquely identify entities (users, groups, …)
  – similar concept to UID/GID in Unix, but unified
  – variable length, numeric values

• Structure
  – SID structure revision number – 48-bit authority value –
    variable number of 32-bit sub-authority
  – Administrator has S-1-5-21-XXX-XXX-XXX-500

• Administrator
  – account similar to the root user in Unix
Impersonation

• Impersonation
  – used to create access tokens with different permissions
  – the Windows equivalent of setuid* calls
  – can be used to elevate or drop access rights
Security Descriptors

- **Security descriptor**
  - security information associated with objects
  - important fields
    - owner SID
    - primary group SID (only used by POSIX)
    - discretionary access control list (DACL) – relevant for access control
    - system access control list (SACL) – relevant for logging

- **Access control list**
  - header + list of access control entries (ACE)
Security Descriptors

• **Access control entry (ACE)**
  – contains a SID (e.g., for user chris)
  – corresponding operations (e.g., write, read)
  – type (that specifies either allow or deny)

• **ACL assignment**
  – complex set of rules:
    either directly set
    or determined via “inheritance” – e.g., from the current directory
    or default taken from access token
Security Descriptors

• Access decision
  – traverse the DACL until
    either all requested permissions are granted, or
    a requested permission is denied
  – this implies that the order of the ACE might matter!
  – typically, deny entries appear first

• Owner of resource always gets right to modify the DACL

• In principle, concepts are more powerful than Unix
  – permissions for many groups can be defined
  – fine-grain control via allow and deny rules possible
Privileges

• Recall that access token also stores privileges

• Privileges
  – not all (security-relevant) operations are associated with objects
    examples: shut down computer, set system time, …
  – other privileges might disable or bypass access control checks
    examples: backup files, debug processes, …

• Super privileges
  – some privileges are so powerful that they basically grant full access
    “Act as part of the OS,” “Debug Program,” “Restore files” …
Authentication

Winlogon process

GINA (Graphical Identification and Authentication)

User Desktop (Shell)

LSA Server (lsass.exe)

Authentication Package (MSV1_0) – LAN Manager 2

SAM (Security Accounts Manager) Server

SAM DB (Registry)
SAM DB

• Stores hashed passwords
  – similar to /etc/passwd (and /etc/shadow)

• Two formats
  – LM (LAN Manager) hash
  – NTLM

• LM hash
  – uses DES to encrypt static string
  – however, a few flaws
    no salt
    splits 14 characters into 2 blocks of 7 characters (hashed separately)
    all characters converted to uppercase (further reduces key space)
SAM DB

- LM hash
  - can be cracked trivially (ophcrack)
  - disabled by default in Vista (or when password > 14 characters)

- NTLM
  - better security (MD5)
  - still no salt, thus effective rainbow table attacks possible
File System

• NT File System (NTFS)
  – successor of FAT (file allocation table) file system
  – better performance, journaling support, quotas
  – supports Windows security features (in particular, access control features)

• Interesting features
  – links (since Vista, even symbolic links :-)
  – alternate data streams (ADS)

• ADS
  – adds additional streams to a file
  – original file size is not modified, and ADS are difficult to identify
  – accessed in the form of filename:streamname (e.g., text.txt:secret)
  – planned to hold meta-data
  – used by malware to hide presence