Special Topics in Security
ECE 5698

Engin Kirda
ek@ccs.neu.edu

Northeastern University

Unix Security

Unix

- Multi-user operating system

- Operating system functionality
  - process management
  - (virtual) memory management
  - file system management
  - I/O management
- Structure
  - operating system kernel
  - user-space programs (daemons, applications, shell)

Unix

- Kernel
  - provides a hardware abstraction layer for user-space programs
  - complete access to all (physical) resources
  - trusted computing base
  - provides services via system calls

- System call
  - performs a transition from user mode to privileged (kernel) mode
  - this crosses the border between two security domains
  - usually implemented with hardware (processor) support
    - processor interrupt
    - x86 call gates

Unix

- Kernel vulnerability
  - usually leads to complete system compromise
  - attacks performed via system calls
  - e.g., a famous one appeared in February 2009, vmsplice
- 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)

Unix

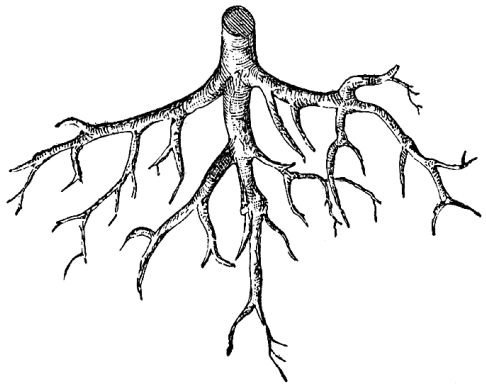
- More Linux vulnerabilities
  - Linux message interface (August 2005, CAN-2005-2490)
  - race condition - `proc` and `prctl` (July 2006, CVE-2006-3626)
  - local privilege escalation - (September 2007, CVE 2007-4573)

- 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
  - implements user-activity
  - entity that executes a given piece of code
  - has its own execution stack, memory pages, and file descriptors table
  - separated from other processes using the virtual memory abstraction
- Thread
  - separate stack and program counter
  - share memory pages and file descriptor table

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
  - Traditional: 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

DEMO, Shadow Passwds…

Group Model

- Users belong to one or more groups
  - primary group (stored in `/etc/password`)
  - additional groups (stored in `/etc/group`)
  - possibility to set group password
  - and become group member with `newgrp`

- `/etc/group`

```
groupname : password : group id : additional users
root:x:0:root
bin:x:1:root,bin,daemon
users:x:100:ek
```

DEMO, groups…

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)
```

Type	r	w	x	s	t
File	read access	write access	execute	suid / sgid inherit id	sticky bit
Directory	list files	insert and remove files	stat / execute files, chdir	new files have dir-gid	files only delete-able by owner

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
  - `preserve` attack (`/usr/lib/preserve` is SUID)
    - called "`/bin/mail`" when `vi` crashed 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

DEMO, Restricted Shell, creating a chroot 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`

DEMO, system() issue…

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