IY5512: Computer Security
(Operating Systems)

Unix and Linux Security

Antony Stone

Unix and Linux Security

- The Unix™ and Linux™ Operating Systems
- Open Source software
- User Accounts & Passwords
- Files & devices
- Ownership & Permissions
- Access control mechanisms
- System auditing & logging
- Extending Linux security

What is (or was) Unix ?

- Operating System for computers
- Multi-user, multi-tasking
- Command-line (text) based (but graphical X-Window system available)
- Originally “power” system for servers and expensive workstations – not PCs
- Available from many different vendors
    - Hewlett-Packard, Sun, IBM, Microsoft.....

What is (or was) Unix ?

- Started in 1969 by Ken Thompson & Dennis Ritchie
    - Still in use 45 years on
- Source Code was freely distributed to those who wanted it (mainly academics)
    - A good example of Open Source Software (although the term wasn't used in those days)

What is Linux ?

- Linus Torvalds' Open Source Unix-clone
  - Started in 1991
  - Available for Intel x86, Alpha AXP, Sun SPARC, Motorola 68000, PowerPC, ARM, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64 and several others.....
- Linux is not derived from AT&T Unix source
- Linux is not Unix, transferred to the PC (Xenix)
- Linux is a completely rewritten O/S kernel
  - And it is, technically, only the kernel

Open Source Software

- Source Code is freely distributed
  - human-readable (vs. Object Code)
  - can be edited, modified, enhanced
  - errors can be found and fixed (or exploited !)
    - by anyone who cares to look
    - by anyone who can understand it
- Most common example of distribution licence: the GNU General Public Licence

  "You can do anything you want with this software except restrict what anyone else can do with it, and you must give back your modifications."

Linux: Origins, Motivations, Outcomes

- Why has Linux become as successful and as widespread as it is ?
  - Can be regarded as “Unix on a PC”
  - Powerful software, cheap hardware
  - Alternative to Microsoft Windows / Apple Mac OS
  - Software is free (no cost, no restrictions)
  - People can see and change the software
    - Add new features
    - Improve what's there already
    - Fix bugs
    - Learn

Linux Distributions

- Linux is the Operating System Kernel
- Applications vary depending on the purpose of the machine
- Different organisations distribute the Linux kernel along with different collections of application software, different ways of installing it, different levels of support...
    - eg: Red Hat, SuSe, Debian, Ubuntu, Gentoo...
- There is still only one Linux kernel

Linux Security

- Linux has the same security model as Unix
  - User IDs, file ownership, access permissions, "root" superuser, password encryption
- Linux has the same security problems as Unix
  - No system audit mechanism, only one superuser, poor and diverse logging, SetUID, TCPwrappers

Linux Security

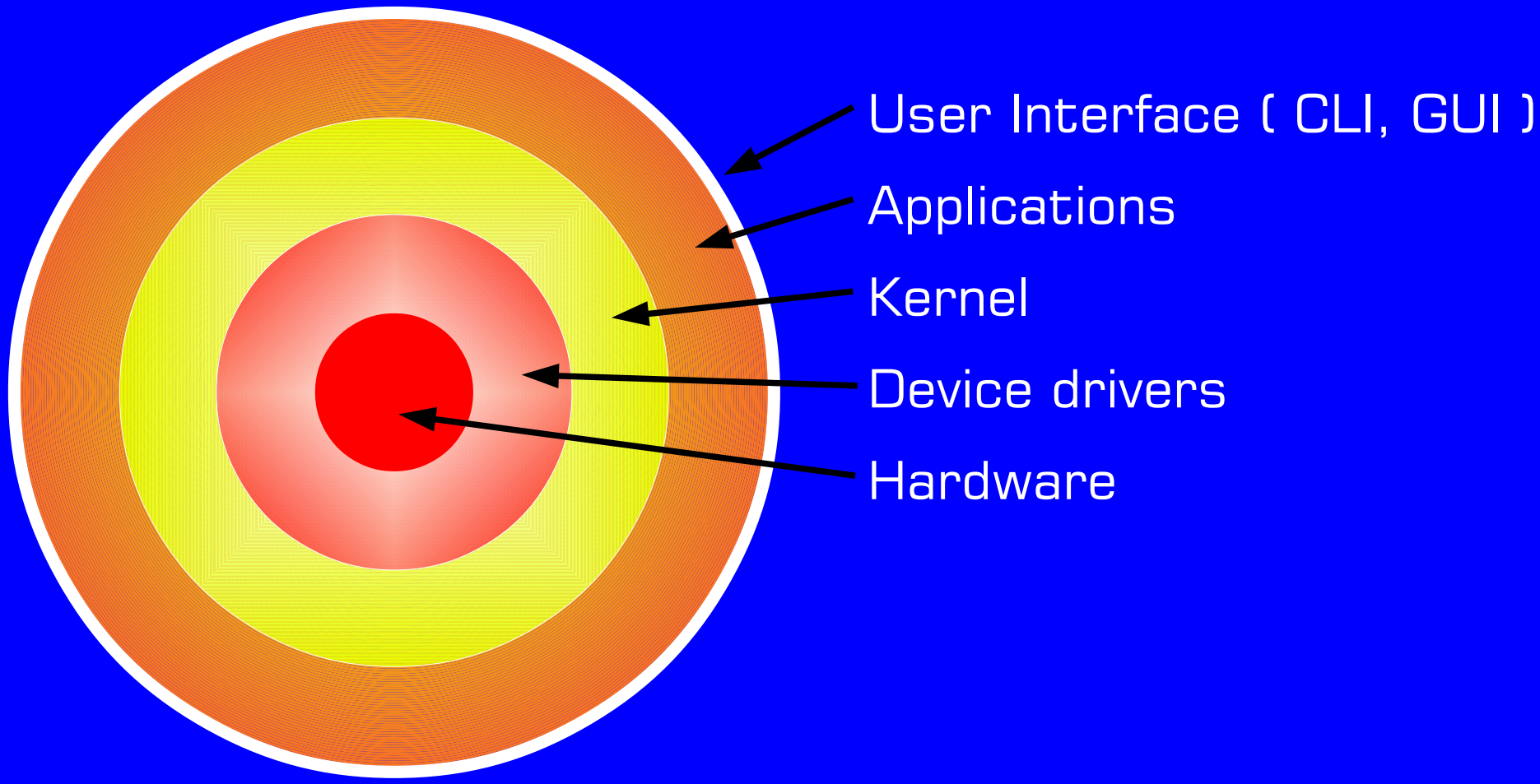
- Why has Linux become so popular ?
  - It's free
    - No cost, no restrictions
  - It's adaptable
    - Take what already works and build on it
  - No vendor lock-in
    - No single vendor
  - People trust it
    - Open Source is visible

The Open Source Advantage

- Linux is very popular with embedded systems manufacturers
  - Small machines with specialised functions
    - Televisions / digital video recorders
    - DSL routers / firewalls / wireless access points
    - Network storage devices
    - Telephones (VoIP, Android)
  - Starting from Linux and adapting to specialised needs is easier than starting from scratch
    - This still inherits the Unix / Linux security model

Operating System Architecture

- Often depicted as an "onion" layered model:



Operating System Architecture

- Each layer interfaces to the layer inside, and to the layer outside (similarly to the OSI networking model)
- Each layer protects the layer/s within from those further out
- eg: Applications cannot directly access hardware
  - must use defined Kernel calls, which use Device Drivers, which then access the hardware

Unix Characteristics

- Designed for accessibility and data sharing
  - remember – mainly academic user-base initially
- Not designed with high security in mind
  - eg: /etc/passwd file is readable by anyone
- Security has been added on later
  - Never the best approach
- Not regarded as a "friendly" Operating System
  - Designed for use by computer experts

Unix Concepts

- Unix was designed as an 'elegant' operating system (programmers' definition)
- Toolbox approach:
  - A command for almost everything
    - cat – concatenate or output files
    - tac – same thing, but output backwards
- Command Line Interface
    - Obscure names for commands
    - Obscure options for different variations

Unix Commands

- Range from the clear & obvious, to the obscure, to the downright strange…
  - date
  - sort
  - reboot
- Not so obvious…
  - cat – to display contents of file
  - grep – to search for strings, eg words
  - touch – to update date / timestamp on file
- And…
  - biff – notification of new email arriving

Unix User Accounts

- All user accounts are recorded in the file /etc/passwd
- The Unix account identifier is the numeric **UID**
  - The user name is associated with the UID merely for human convenience
- Every account also has a group identifier **GID**
- /etc/passwd file is needed by a great many applications in order to link username to UID
- /etc/passwd is therefore "world readable"

Contents of /etc/passwd

```
root:x:0:0::/root:/bin/bash
operator:x:11:0:operator:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
bin:x:1:1:bin:/bin:
ftp:x:404:1::/home/ftp:/bin/bash
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
man:x:13:15:man:/usr/man:
games:x:12:100:games:/usr/games:
guest:x:405:100:guest:/dev/null:/dev/null
nobody:x:-2:100:nobody:/dev/null:
godfrey:x:1000:100:Godfrey Rock:/home/godfrey:/bin/bash
antony:x:1001:100:Antony Stone:/home/antony:/bin/bash
general:x:1002:100:Samba Account:/home/general:/bin/bash
```

Contents of /etc/passwd

- 7 fields separated by colons :
  - Username
  - Password
  - UID
  - GID
  - Comment
  - Home directory
  - Login shell

Contents of /etc/passwd

```
root:x:0:0::/root:/bin/bash
operator:x:11:0:operator:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
bin:x:1:1:bin:/bin:
ftp:x:404:1::/home/ftp:/bin/bash
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
man:x:13:15:man:/usr/man:
games:x:12:100:games:/usr/games:
guest:x:405:100:guest:/dev/null:/dev/null
nobody:x:-2:100:nobody:/dev/null:
godfrey:x:1000:100:Godfrey Rock:/home/godfrey:/bin/bash
antony:x:1001:100:Antony Stone:/home/antony:/bin/bash
general:x:1002:100:Samba Account:/home/general:/bin/bash
```

Username: antony
Password: x
UID: 1001
GID: 100
Comment: Antony Stone
Home directory: /home/antony
Login shell: /bin/bash

Contents of /etc/shadow

```
root:$1$SBWwi99V$tDWSSzOxpUtQe5CoM2lH21:11790:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
mysql:*:9797:0:::::
gdm:*:9797:0:::::
nobody:*:9797:0:::::
antony:$1$/R.xwrNZ$/R1Z8nLeukBKKxAHfmRVH1:11818:0:99999:7:::
godfrey:$1$/7e9xc8f$38Z9MwSVPV02USdQX4iKY/:11966:0:99999:7:::
general:$1$1SLPxHzB$XXhc3/xLeqZGxdQBq/sSL/:11905:0:99999:7:::
```

Contents of /etc/shadow

- 9 fields separated by colons :
  - Username
  - Password hash
  - Date of last password change
  - Days until change allowed
  - Days before change required
  - Days warning for expiration
  - Days before account inactive
  - Date when account expires
  - Reserved for future use

Contents of /etc/shadow

```
root:$1$SBWwi99V$tDWSSzOxpUtQe5CoM2lH21:11790:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
mysql:*:9797:0:::::
gdm:*:9797:0:::::
nobody:*:9797:0:::::
antony:$1$/R.xwrNZ$/R1Z8nLeukBKKxAHfmRVH1:11818:0:99999:7:::
godfrey:$1$/7e9xc8f$38Z9MwSVPV02USdQX4iKY/:11966:0:99999:7:::
general:$1$1SLPxHzB$XXhc3/xLeqZGxdQBq/sSL/:11905:0:99999:7:::
```

Username: antony
Password: $1
  salt: $/R.xwrNZ
  hash: $/R1Z8nLeukBKKxAHfmRVH1
Date of last password change: 11818
Days until change allowed: 0
Days before change required: 99999
Days warning for expiration: 7
Days before account inactive: blank
Date when account expires: blank

Unix Password Encryption

- Old system (until c.1995)
  - So, the first 25 years of Unix...
  - Single DES
  - Max. 8-character passwords (56-bit key)
  - 64-bit encrypted value stored in /etc/passwd
    - (World-readable)
  - Vulnerable to 2 types of dictionary attack
    - Targeted system
    - Any system

Unix Password Encryption

- New system (since c.1995)
  - MD-5 instead of DES
  - Any length passwords
  - 128-bit encrypted value stored in /etc/shadow
    - (Root access only)
  - Dictionary attack virtually useless
    - Have to have root access to see encrypted passwords
    - Still a problem if people use same password on multiple systems

Contents of /etc/group

- 4 fields separated by colons :
  - Group name
  - Group password
  - GID
  - Username list (separated by commas)

Contents of /etc/group

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:man
users::100:games
nogroup::-2:
```

UID=0 (zero)

- The "root" user (name is by convention)
- Superuser
- Unrestricted access to entire system
  - Can create (and delete) other user accounts
  - Can access (and change or delete) any files anywhere on the system, owned by anyone
  - Can destroy the entire system with a single command if not careful !
    - rm -rf /*
      - Note that Unix has no undelete command...

UID=0 (zero)

- All-powerful user – therefore dangerous
- Cannot generally log in remotely as root
  - Login as normal user, then use “su” or “sudo”
  - Provides (meagre) audit trail
  - You should stop being root as soon as possible
    - Protect the system from your own mistakes
- Major disadvantage with Unix root user – there is only one superuser: UID=0
    - Unlike Windows Administrators

Unix root

- A note about the word “root” in Unix
  - Two distinctly different meanings:
- The all-powerful super-user UID=0
  - The name is merely a convention
  - Changing the word “root” in /etc/passwd and /etc/shadow makes no difference to functionality
- The base directory of the file system /

The Unix File System

- All files (on all drives) appear in a single directory tree starting from /
- Different devices can be “mounted” at different locations within the directory tree
  - eg: First hard disk partition = /
  - Second hard disk partition = /home
- Remote network “drives” are mounted in the same way within the single directory tree
  - eg: /home/antony/fileserver
- Windows-style drive letters do not exist

The Unix File System

- Commonly-seen directories:

/bin	Binaries (programs)
/boot	System boot information
/dev	Devices
/etc	Miscellaneous files
/home	Users' home directories
/lib	System libraries
/root	Root user's home directory
/sbin	System binaries
/tmp	Temporary filespace
/usr	User binaries etc
/var	Variable files eg: system logs

The Unix File System

- Devices are represented as special names within the file system, eg:
  - Console (display) /dev/console
  - keyboard /dev/kbd
  - mouse /dev/psaux
  - serial port /dev/ttyS1
  - USB port /dev/ttyUSB0
  - sound card /dev/dsp
  - hard disk drive /dev/sda

Unix Files & Devices

- Why treat devices the same as files ?
- Programming elegance – eg: consistency for commands:

```
echo “Hello World.”
echo “Hello World.” >myfile
echo “Hello World.” >/home/antony/myfile
echo “Hello World.” >/dev/tty1
echo “Hello World.” >/dev/lp0
echo “Hello World.” >/dev/ttyS1
echo “Hello World.” >/dev/sda1
```

Unix Permissions

- Every file (and device) on a Unix system has an owner and a group
- Owner is a username on the system
  - Technically, a numeric UID
- Group is a groupname on the system
  - Technically, a numeric GID
- The Owner need not be a member of the Group
  - The owner and the group for a file are entirely independent of each other

Unix Permissions

- Every file (and device) on a Unix system has a set of permissions associated with it
- Three types of permission:
  - read
  - write
  - execute
- Permissions are granted in three categories:
  - user (owner)
  - group
  - other

Unix Permissions

- That's it
- Extremely simple security model
- No Access Control Lists
  - (But can be added)
- Privilege is pretty much “All or nothing”
  - Root vs. standard user
  - Root can do anything

Unix Permissions

- The "mode" of a file determines which user/s are permitted different types of access
- The owner of a file (or the "root" superuser) can always change the mode of a file
  - even if the mode denies access to the file for the owner (which is perfectly possible !)
- Remember that a user can be a member of more than one group
  - a file has exactly one owner and one group

Unix Permissions (important slide)

- Permissions apply to the most specific categorisation of the user requesting access
  - If the user is the owner of the file, then the Owner permissions only are applied
  - Otherwise, if the user is a member of the file's group, then the Group permissions are applied
  - Otherwise the Other permissions are applied
    - If the user requesting access is the owner, then the Group and Other permissions are irrelevant

Unix Permissions

- File mode usually written as: rwxrwxrwx
- Can be expressed numerically, in octal
- user, group, other
- read, write, execute
- eg:

```
-rw-r-----    antony   users    11264 Jun 23 13:45 Comments.odt
-rw-r--r--    antony   users    13156 Sep 25 00:05 Graphic.gif
-rw-rw-r--    antony   users     7680 Jun 23 13:43 Notes.txt
-r--r-----    antony   users     9431 Aug 19 18:04 Slide.pdf
```

Unix Permissions

```
-rw-r-----  antony  users  11264 Jun 23 13:45 Comments.odt
```

- Owner = antony
- Group = users
  - User permissions = rw-
  - Group permissions = r--
  - Other permissions = ---
- User “antony” can read and write file
- Any user in group “users” can read file
- Any other user cannot access file

Unix Permissions

- Execute permission
- Unix filenames do not use extensions such as .exe .com .bat .pif to indicate whether they contain executable content
- Execute permission bit is used to determine whether a file can be run as a program
- Unix allows a program to be executed without read permission on the file
    - A user can run a command but can't read the file

Unix Permissions

- How to stop yourself from being able to read a file which you own:

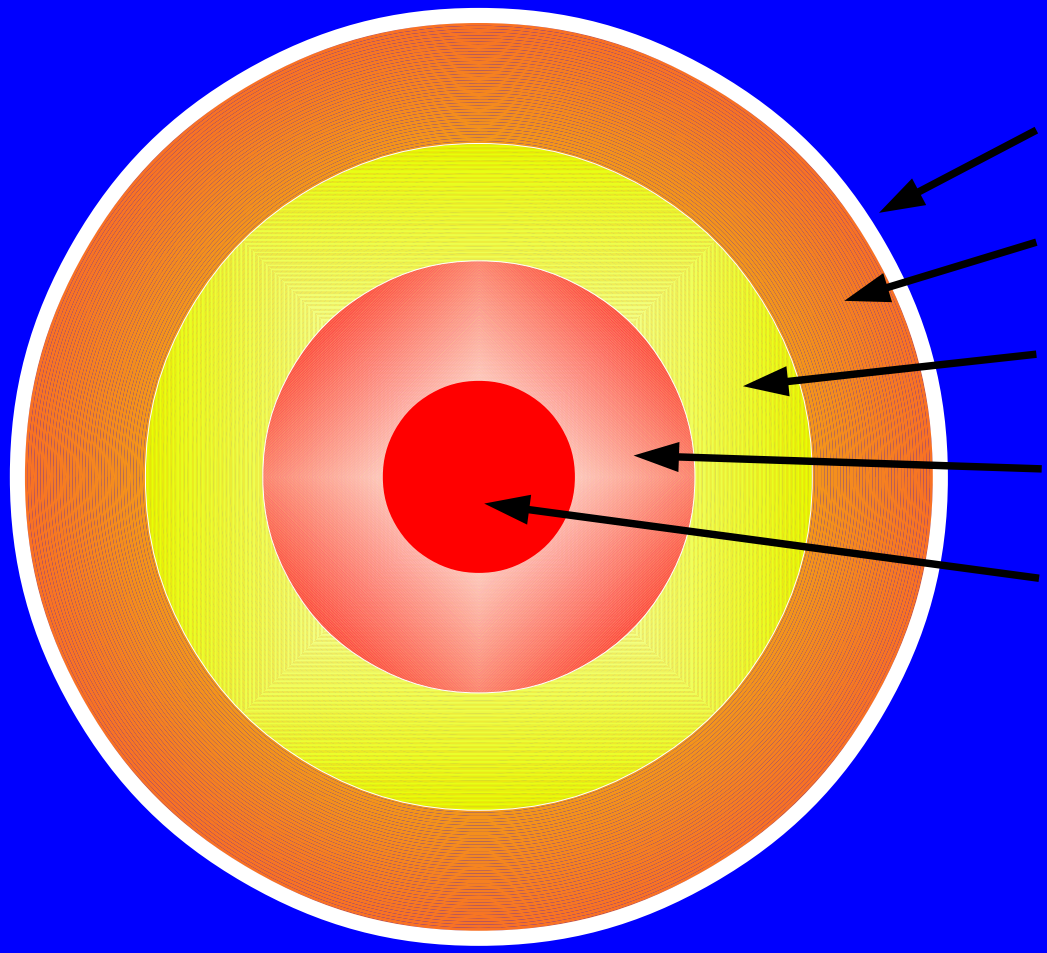
```
$ ls -l myfile
-rw-rw-rw-   antony   users    myfile
$ cat myfile
The contents of my file....
$ chmod u-r myfile
$ ls -l myfile
--w-rw-rw-   antony   users    myfile
$ cat myfile
cat: myfile: Permission denied
$
```

Unix Access Control Mechanisms

- Unix does not use Access Control Lists
  - But these can be added on top
- All Unix file security is based on permissions
- The root user with UID=0 can do anything
- Therefore attackers often want to "become root"
- The Unix kernel enforces file permissions
- Recall the onion model

Unix Access Control Mechanisms

- Layers of security from user to hardware (data)



User Interface (CLI, GUI)

Applications

Kernel

Device drivers

Hardware (data)

- disks, network, memory

Unix Access Control Mechanisms

- Unprivileged applications (those running under UID & GID > 0) cannot access privileged data
  - "Privileged data" includes hardware devices
- UID = 0 applications can access privileged data
  - may be made to display it, transmit it, copy it, modify it, delete it.....
  - How to become root (without a password)?
    - buffer overflow exploits
    - application insecurities

Application Insecurities

- Recent Open Source security problems
  - Heartbleed (April 2014)
    - Failure of boundary-checking in OpenSSL library
    - Used by millions of secure websites worldwide
  - Truecrypt (May 2014)
    - Popular disk encryption mechanism, abandoned by the developers, “may contain unfixed security flaws”

Application Insecurities

- Shellshock (September 2014)
  - External input can be interpreted as commands
  - Several different vulnerabilities / exploits
    - Including HTTP requests to web servers
  - Faulty code has been in place since 1989 (25 years)
- OpenSSL (June 2015)
  - Certificate chain checking error
  - Allows attacker to create fake Certificate Authority
- Truecrypt update (September 2015)
  - Privilege escalation allows local users to access encrypted data

Application Insecurities

- Open Source is not a magic solution for security problems
  - Just because anyone can read the code, find the bugs, and fix them, doesn't automatically mean anyone does
  - Some people may read the code, find the bugs, and create the exploits
  - A lot of Open Source is volunteer-developed
    - Is anyone still involved to distribute updates?
    - Who provides resources to find & fix problems?

Application Insecurities

- Closed Source is not a magic solution for security problems
  - Just because almost no-one can read the code and find the bugs, doesn't automatically mean they don't get found and exploited
    - MS Windows has a long history of security problems
      - Most common systems, therefore popular target
  - End users are dependent on a single supplier to react, issue updates, provide support
    - Supplier may no longer exist
    - Product may be out of support (eg: Windows XP)

Damage Limitation

- Principle of Least Privilege
  - Applications should have only sufficient privilege for their needs at the time
- It's common to have a user “nobody” under Unix
  - Not allowed to log in
    - No password (not blank; actually non-existent)
    - No login shell
  - Used as unprivileged user for background applications (daemons)

Damage Limitation

- Principle of Least Privilege
  - Applications should have only sufficient privilege for their needs at the time
- If you use su or sudo, exit as soon as possible
- SetUID utilities are especially sensitive
- Background processes (daemons) usually have to start with root privilege
  - Keep that code small and clean
  - Drop root privilege as soon as possible

Unix Permissions Again

- Example of SetUID usage
  - Dilemma:
    - /etc/shadow file must be protected from ordinary users & applications
    - Every user's password is stored (encrypted) in /etc/shadow
    - How does a user change their own password ?
    - What stops a user changing another user's password ?

Unix Permissions Again

- Special Execute permission – SetUID
- A SetUID program will execute with the privileges of its owner, no matter which user runs the command
- Hence a normal user can change the contents of the (secure) /etc/shadow file by using the passwd command

```
-rw-------    root    root    /etc/shadow
-r-s--x--x    root    root    /usr/bin/passwd
```

Unix Permissions Again

- SetUID programs are inherently dangerous
- SetUID does not necessarily mean UID 0, although this is nearly always the case
    - Unix security is essentially “root or non-root”
- If SetUID programs contain any vulnerabilities, these would almost certainly result in a root privilege exploit
    - Then the entire security model is gone
    - Root can do anything

More Damage Limitation

- Unix has a “chroot” command
  - Defines a subdirectory somewhere in the file system as the base “root” directory for a command
  - eg: /home/ftp
  - The application running inside the “chroot jail” cannot access anything outside
    - Data is separated into compartments
    - Different applications cannot see each other's data

Unix Auditing

- Unix does not have a satisfactory auditing capability
- Not even a satisfactory alerting capability
- Unix maintains several system log files, eg:
  - /var/log/messages
  - /var/log/syslog
  - /var/log/debug
- Mainly of technical informational interest

Unix Logging

- Many applications maintain their own log files
  - eg: Apache web server often logs all requests
- Many applications & commands log events of varying significance
  - eg: “su” command logs username and new ID
  - ftp daemons commonly log file transfers
  - pop3 daemons commonly log connections
- Some potentially useful information does not get logged, or is done trivially
  - eg: the commands which are issued by UID 0

Unix Logging

- An example of good intentions, poor implementation:
- "TCPwrappers" used by many Unix network applications
  - A "reverse lookup" to discover the identity of the user making a request across a network
    - "I want to connect to this service."
    - "Who are you ?"
    - "I am Antony."
    - "Okay - proceed."

Unix Logging

- TCPwrappers log responses in system log file
- But, what if the requesting system doesn't support the IDENT lookup ?
  - "I want to connect to this service."
  - "Who are you ?"
  - ....pause....
  - "Okay, you can connect anyway."
  - Same thing happens if the response is "I'm not telling you."

Unix Logging

- Information overload.....
  - Too much data
  - Too much is trivial
  - Too spread out
- Log file analysers
  - Real-time
  - Offline / batch
- Log file archiving
  - Remote logging across network

Unix Logging

- What use are log files anyway ?
  - Normal system operation – why bother ?
  - Abnormal system operation – can you find the information you need ?
  - System under attack – is anything useful logged anyway ?
  - System after attack – is the information still there ?
  - Audit trail – can you tell who did what ?
  - Remember – there is only one superuser: UID=0
    - No matter how many people have access

Unix Security

- How can we make a Unix system secure ?
- Operating System "Hardening"
  - Remove unnecessary commands
  - Disable / remove unnecessary services
  - Tighten file system permissions
  - Check user account password strength
  - Vet all SetUID programs
  - Apply security patches / updates

Physical Security

- The Operating System can only protect a machine if the Operating System is running !
  - Everyone has access to Linux these days
    - Years ago, Unix was only for big, expensive systems, and used proprietary disk formats
    - Now, Linux will fit on a CD-Rom or a USB stick
      - Including network support
  - Linux can read nearly all system file formats:
    - Linux, BSD, AIX, Xenix, QNX, CP/M, Novell, Microsoft FAT16, FAT32, NTFS, BeOS . . . .

Physical Security

- An Operating System cannot stop someone walking up to a machine, rebooting from their own CD or USB stick, and having total access to the system...

```
# mount  /dev/sda1  /target
# cat  /target/etc/shadow

# echo "r00t::0:0::/root:/bin/bash" >>/target/etc/passwd
```

- Hardware is the lowest layer
  - Physical security is still important !

Extending Linux Security

- "Adequate security cannot be provided in applications with the existing security mechanisms of mainstream operating systems."
    - NSA, 2000

Extending Linux Security

- Security Enhanced Linux (SE Linux)
  - Developed by the NSA...
  - Open Source (because it's in the Linux kernel)
- AppArmor
  - Developed by Immunix / Novell / Canonical
  - Also Open Source
- Docker
  - And other “container” implementations
  - Data isolation

SE Linux

- SE Linux provides:
  - Mandatory Access Controls
  - Fine-grained Permissions
  - Minimisation of Privileges
- Standard permissions system remains
- SE Linux support is included in the standard Linux 2.6 and 3.x kernels, and all main Linux distributions

SE Linux

- Based on a combination of:
  - IBAC - Identity Based Access Control
  - RBAC - Role Based Access Control
  - TE - Type Enforcement
- Much more “fine-grained” than the standard Unix security permissions model
- Similar to Access Control Lists, but takes the concept much further

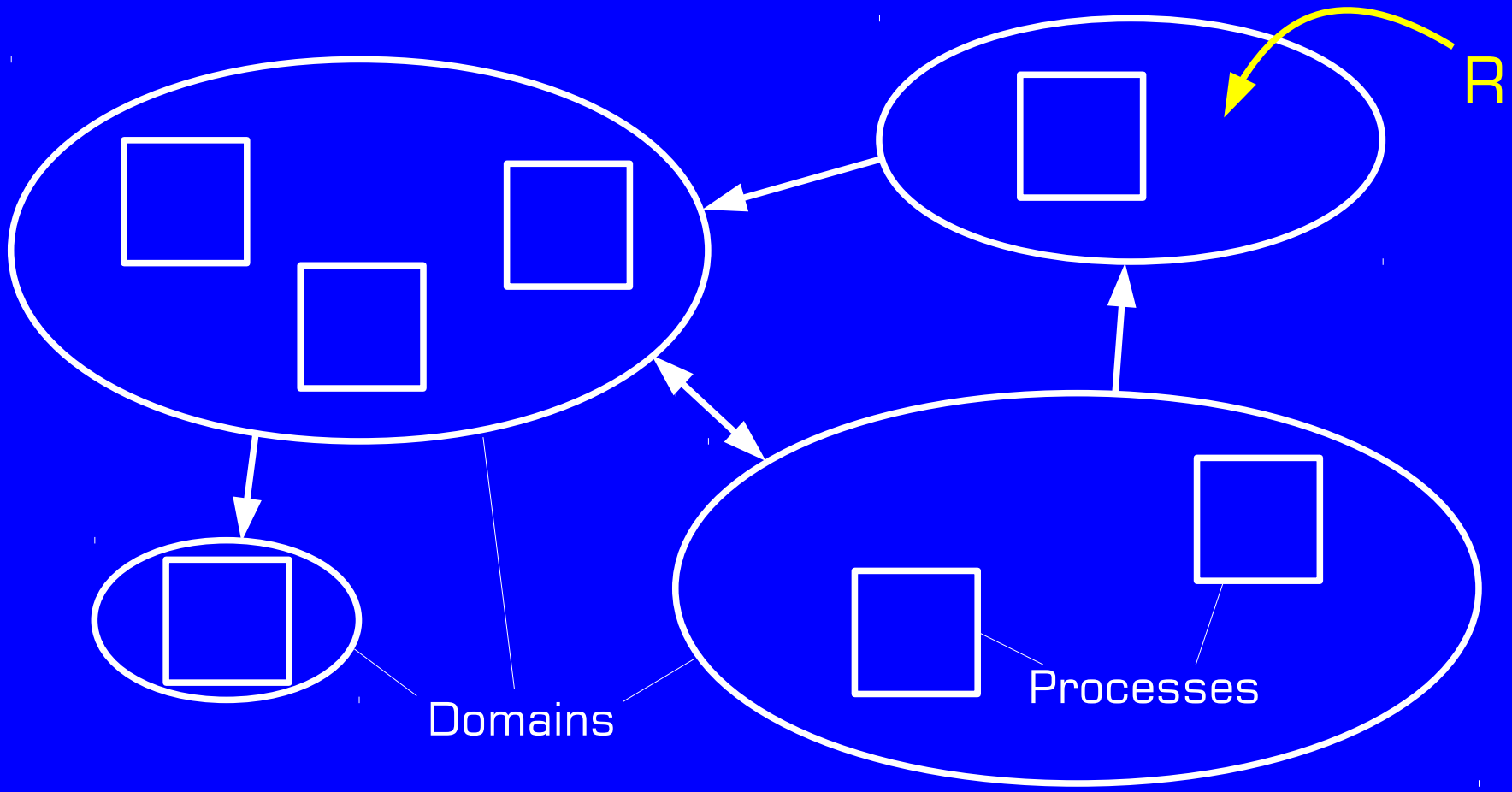
SE Linux - principles

- Processes are separated by independent Domains
    - No communication between domains is allowed unless a rule permits it
- There is no single “root” superuser
    - Any user can run any process which the rules permit them to
- Every system process has an identity
    - Independent of Linux UIDs

SE Linux - principles

- Each user's actions are restricted by RBAC
  - Users are assigned a set of roles
  - SE Linux does not assign permissions to roles
  - SE Linux specifies domains which may be 'entered' by roles
  - permissions are associated with domains by the Type Enforcement configuration
- RBAC means that several users can have (selected) root privileges

SE Linux - principles



SE Linux - principles

- Type Enforcement is used to implement fine-grained access controls
- Each system process exists within a domain
- System objects are associated with types
- Configuration files specify:
  - what types (of objects) can be used to enter domains
  - what communication is permitted between domains
    - Network firewall analogy is useful

AppArmor

- First released 1998
  - Before SE Linux
  - Developed by Immunix (commercial Linux distribution)
  - Acquired by Novell 2005-2007
  - Since 2009, maintained by Canonical
- Open Source (GPL)
- Integrated into Linux kernel since Oct 2010
- Similar idea to SE Linux, but simpler model

AppArmor

- Based on Mandatory Access Controls
- Also contains Learning Mode
  - Define applications, let them run normally
  - System creates rulesets for secure operation
- Easier to administer than SE Linux
  - Fewer new concepts
- Does not require special file system support
  - Unlike SE Linux

Docker (and similar systems)

- "Container" concept
  - Applications run inside containers
  - Containers exist inside the Operating System
- Lighter on resources than full virtualisation
  - Only one Operating System for many containers
- More powerful management than chroot jails
  - Restrict communications to / from:
    - Disks
    - Network
    - Other containers

Docker (and similar systems)

- Not specifically designed for security
  - But has several features which help
    - Isolation of applications and data
    - Control of communications
- Primary focus is making resources available cheaply / quickly
  - Not as complex as virtualisation
  - Cloud computing

Extending Linux Security

- SE Linux
  - Developed by the NSA
- AppArmor
  - Much more openly-developed project
- Both based on Linux kernel enhancements
- Better security models than standard Unix
- RBAC / MAC comparable to the Windows security model

Extending Linux Security

- Docker
  - And similar container implementations
- Not specifically designed for security
  - Performance and application isolation
  - "Halfway house" between chroot jails and full machine virtualisation
- Most container implementations manage:
  - Disk / memory / CPU allocations
  - Network / file system isolation