

UNIX & Security

Benjamin Brewster

Except as noted, all images copyrighted with Creative Commons licenses,
with attributions given whenever available

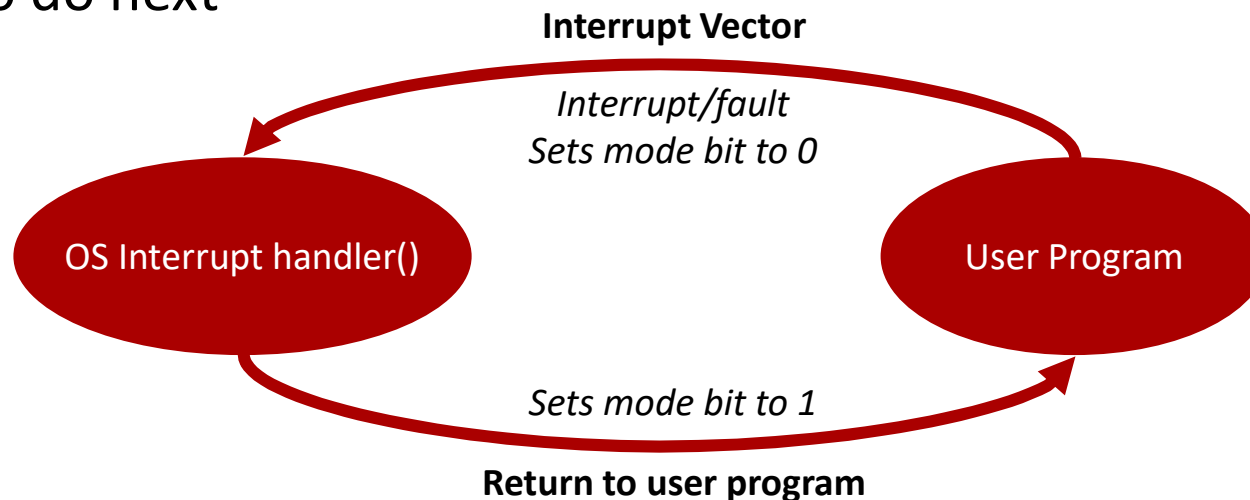
Dual-Mode Operation

- Sharing system resources requires the operating system to ensure that a program cannot arbitrarily interfere with other programs
- The hardware itself provides support to differentiate between at least two modes of operations:
 - User mode: execution done on behalf of a user
 - Monitor mode (also supervisor mode or system mode): execution done on behalf of the operating system
- Privileged instructions can be issued only in monitor mode



Dual-Mode Operation

- The *mode bit* is added to computer hardware to indicate the current mode: monitor (0) or user (1)
- When an interrupt or fault occurs, the hardware switches to monitor mode by following the address, stored in the *interrupt vector*, to the *interrupt handler* function in the OS; this handler will let the OS decide what to do next



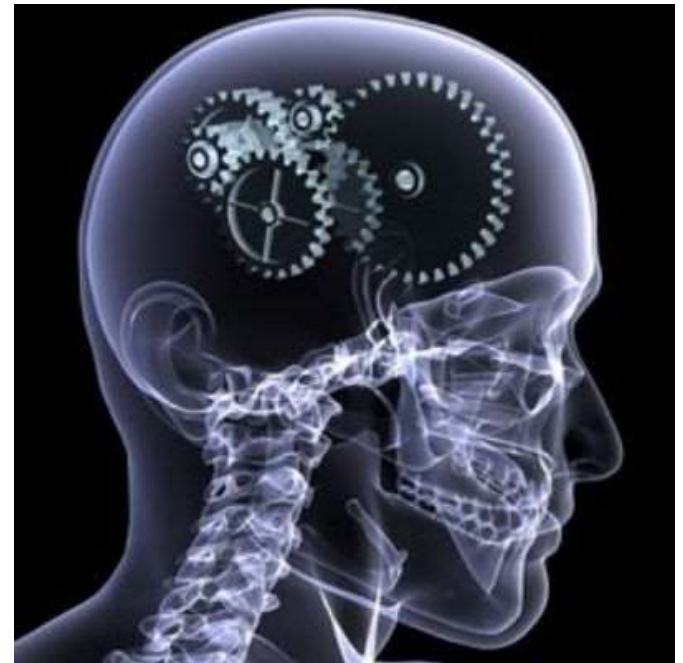
I/O Protection

- All I/O instructions (`read()`, `write()`, `send()`, `recv()`, `fgets()`, `putc()`, etc.) are privileged instructions
- Because: the OS must ensure that a user program could never gain control of the computer in monitor mode by storing a new address in the interrupt vector



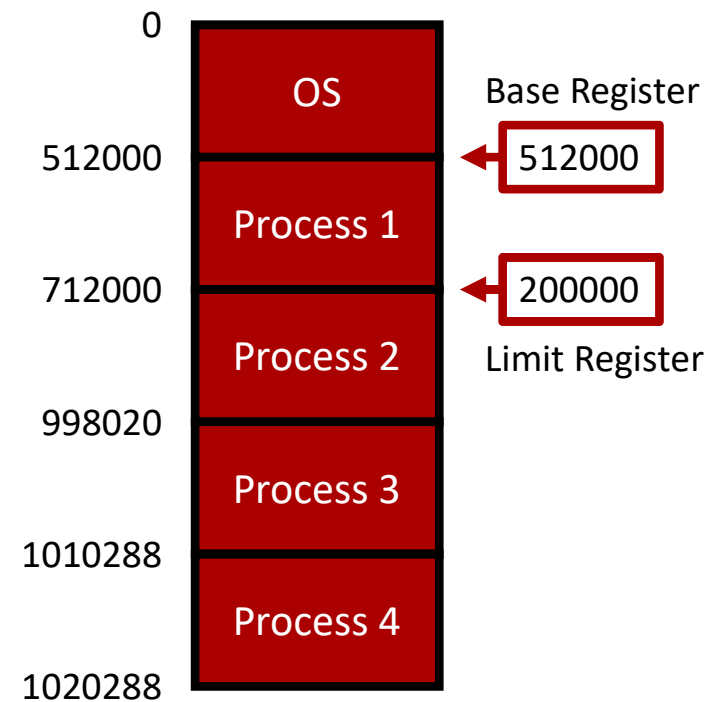
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt handler function
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
 - **Base register** – holds the smallest legal physical memory address.
 - **Limit register** – contains the size of the range
- Memory outside the defined range is protected



Memory Protection

- The base and limit registers define a logical address space, which is virtualized for the process to start at address 0
- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory
- Obviously, the load instructions for the base and limit registers are privileged instructions



CPU Protection

- If the CPU is executing program instructions one after the next, how does the OS retain control?
- A timer interrupts the control flow after a specified period to ensure that the operating system has a chance to determine what to do
 - Timer is decremented every clock tick
 - When timer reaches the value 0, the interrupt vector is followed to the interrupt handler
- Timer commonly used to implement time sharing
- Also used to compute the current time
- “Load-timer” is a privileged instruction



General-System Architecture

- Given the I/O instructions are privileged, how does the user program perform I/O?
- With a *system call*: the method used by a process to request action by the operating system
 - Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to monitor mode
 - The monitor verifies that the parameters are correct and legal, executes the request, and returns control to the program instruction immediately following the system call




User Account Rights Protect Files

- User files are protected from other users by defining access based on user accounts
- If you are logged in as an account with access (e.g., you're the owner, or a group owner), you can manipulate the file:
 - `chmod`
 - `vim`
 - `touch`
 - `rm`
 - etc.



Acting as a Different User - Pretexting

- If you want to temporarily act as a different user (but stay logged on as yourself), you can use the `su` command:
 - `su yoog`
- You can also execute just one action with the `sudo` command:
 - `sudo -u yoog rm -rf ~/yoogFiles/*`
- These commands change your effective user and/or group IDs, all of which can be displayed with the `id` command

The root User Account

- Most UNIX systems have a super-user account, typically called root, which has permissions to do anything
 - `su root`
 - `sudo -u root pkill -u brewsteb`
- As root, you can change file ownership, change limits on how many processes users can run at once, add and delete user accounts, and many other things
- It is generally considered bad form to stay logged-in to root itself - it's preferred that you make use of `sudo` to make changes



SUID, SGID

- Each executable has two security bits associated with it: SUID, and SGID
 - If SUID is set, the executable runs with effective user ID of the *owner* of the file
 - If SGID is set, the executable runs with effective user ID of the *group owner* of the file
- This is different from before – we're now talking about specific executables that have bits that enable them to run as different users
 - As opposed to *being* a different user, and then running programs, as `su` and `sudo` allow



Changing SUID Example

```
$ which ping
```

```
/bin/ping
```

```
$ ls -pla /bin/ping
```

```
-rwsr-xr-x. 1 root root 38264 May 10 2016 /bin/ping
```

```
$ chmod u-s /bin/ping
```

```
chmod: changing permissions of `/bin/ping': Operation not permitted
```

Can't change the permissions
of a file I don't own

```
$ ls -pla junk.test
```

```
-rw-rw----. 1 brewsteb upg57541 332 Nov 17 09:47 junk.test
```

```
$ chmod u+s junk.test
```

```
$ ls -pla junk.test
```

```
-rwSrW----. 1 brewsteb upg57541 332 Nov 17 09:47 junk.test
```

Capital 'S' means that the SUID
bit is set, but user execute is not

```
$ chmod u+x junk.test
```

```
$ ls -pla junk.test
```

```
-rwsrw----. 1 brewsteb upg57541 332 Nov 17 09:47 junk.test
```

Changes to lower case 's' now
that SUID is set

chmod Revisited

- It turns out that there are twelve mode bits:
 - 4000 - Setuid on execution
 - 2000 - setgid on execution
 - 1000 - set sticky bit
 - 0400 - read by owner
 - 0200 - write by owner
 - 0100 - execute by owner
 - 0040 - read by group
 - 0020 - wr
 - 0010 - execute by group
 - 0004 - read by others
 - 0002 - write by others
 - 0001 - execute by others



Why SUID Matters

- What if you replace the contents of the real `ping`, which has SUID set and is owned by root, with your own code?
- It would have the same permissions (owned by root), but could do anything you want to the system



Why SUID Matters

- What happens when you set the SUID bit on your own executables?
- They would still be owned by you, and thus would run as you
 - Since you're not root this isn't very interesting
- Can you give your custom executable to root?
- No – this is specifically why you have to *be* logged in as root to change file ownership!
 - `chown` doesn't work unless you're root
 - `chgrp` don't work unless you are a member of that group



Strongest Forms of Security

- The strongest forms of security involve network and physical isolation, but these seriously limit utility
- If you do grant physical access to your computer - even disabling local login access - you still have to worry about:
 - Bootable devices (live CDs, flash drives, etc.) can boot a different OS that can access the hard drive of your computer
 - Hard drive could be stolen and read
 - Reading link-level NIC lights, keyboard EM
- With local logins, passwords = pain



Actual Password Security... is a Pain in the Neck

- Don't let users write them down
- Age the passwords
- Enforce stronger (but more annoying) passwords
 - 1337: @nt3@t3|2
 - random: Z1#3s8u*h
 - long: Ho\\doYouTypeMeF@st
- Restrict use of previous passwords
- Password dictionary check



Password Security

- Longer is better than more complicated
 - Lower case letters = 26 possibilities per character
 - Upper case letters = 26 possibilities per character
 - Numbers = 10 possibilities per character
 - Special Characters = 30 possibilities per character
 - Any given character could be 1 of 92 choices
 - There are then 92^8 8-character passwords:
 - $92 \times 92 \times 92 \times 92 \times 92 \times 92 \times 92 \times 92 = 92^8$
 - $92^8 = 5.1 \times 10^{15} = 5,132,188,731,375,616$

Password Security

- Longer is better than more complicated
 - $92^8 = 5.1 \times 10^{15} = 5,132,188,731,375,616$
- Using just lower case letters:
 - $26^8 = 2.0 \times 10^{11} = 208,827,064,576$
- A 12 character, lower-case password:
 - $26^{12} = 9.5 \times 10^{16} = 95,428,956,661,682,176$

Password Security

- Which is easier to remember:
 - TR0m&on3
 - ihavetwoarms
- Which are you more likely to write down?
- FYI, 4 common words are important in the example above
 - See xkcd's excellent correct horse battery staple comic:

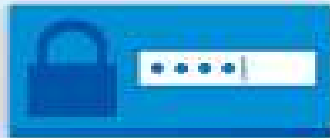
<https://xkcd.com/936/>

Most common
passwords
recovered from
hacked data
dumps



WORST

PASSWORDS OF 2014



- 1 123456
- 2 password
- 3 12345
- 4 12345678
- 5 qwerty
- 6 123456789
- 7 1234
- 8 baseball
- 9 dragon
- 10 football

Login Failures

- What happens if you don't lock a user account if too many failures happen?
 - A account can be brute forced by guessing possibilities
- Passwords are generated with the sausage model (one-way):
 - username: UserBob
 - password: 123456 -> *hashes to* -> a3R7nito5fo%r
- Store the pair UserBob / a3R7nito5fo%r
- This encrypted pair is public knowledge, but the encryption method is one-way



Password Encryption

- If anyone knew how to reverse the password method, then:
 - a3R7nito5fo%r -> *comes from* -> 123456
- Fortunately it is very hard to crack the one-way encryption
- Problem: why is storing the password file publicly dangerous, and why is having a large encrypted password file stolen a problem?
 - A dictionary can be built of encryptions by turning the crank sequentially:
 - 123454 = JoF9#\$94(4k9!
 - 123455 = fj49#mc903#0Q
 - **123456 = a3R7nito5fo%r**
 - 123457 = h9^wehf9*3xd9



Monitoring and Logs



- With all of the insecure protocols still in use (telnet, FTP), keep a tight eye on everything with log files:
 - Network
 - Account login/logout
 - Program usage
 - File access
 - Security checks
 - etc.

Getting Root Access when you're not supposed to have it...

- Try the front door first:

ACCOUNT: PASSWORD

- **root: root**
- sys: sys / system / bin
- bin: sys / bin
- **mountfsys: mountfsys**
- adm: adm
- uucp: uucp
- nuucp: anon
- anon: anon
- user: user
- games: games
- **install: install**
- demo: demo
- **umountfsys: umountfsys**
- **sync: sync**
- admin: admin
- guest: guest
- daemon: daemon



Getting Root Access when you're not supposed to have it...

- Assuming social engineering didn't work, you'll have to use fancy stuff:
 - Port scans + port/program insecurities
 - Buffer overflows (with system access)
 - Boot hacking (with physical access)
- Why are we talking about this stuff?
 - So you can protect yourself against it



Breaking Into Windows

- Boot off of the installation media (Windows Server 2008 DVD, here)

What to know before installing Windows

Repair your computer

Copyright © 2009 Microsoft Corporation. All rights reserved.

<http://www.howtogeek.com/106333/how-to-reset-your-forgotten-domain-admin-password-on-server-2008-r2/>

Breaking Into Windows

- Click here...



Breaking Into Windows

- Enter these two commands...



```
Administrator: X:\windows\system32\cmd.exe
X:\Sources>MOVE C:\Windows\System32\Utilman.exe C:\Windows\System32\Utilman.exe.bak
1 file(s) moved.
X:\Sources>_
```



```
Administrator: X:\windows\system32\cmd.exe
X:\Sources>COPY C:\Windows\System32\cmd.exe C:\Windows\System32\Utilman.exe
1 file(s) copied.
X:\Sources>_
```

Breaking Into Windows

- Reboot without CD, booting normally into Windows on the hard drive, then click here:



- Instead of accessibility, you get a privileged prompt! Change the password like this...

```
Administrator: C:\Windows\system32\utilman.exe
C:\Windows\system32>net user administrator *
Type a password for the user:
Retype the password to confirm:
The command completed successfully.
C:\Windows\system32>
```

Breaking Into Windows

- Log in using the new password!
- Remember to put the files back where you got them from
- Works in Windows 7, 8, 8.1, 10, and Server 2012, too!
- Why not create a few new local user accounts of your own, while you're in there?
- Q: Why can't we create domain accounts?



Password Annihilator - with Physical Access

- Reset, change, or blank out any Windows password by booting from a flash drive or CD:
 - <http://pogostick.net/~pnh/ntpasswd/>

```
*****
* Windows NT/2k/XP/Vista Change Password / Registry Editor / Boot CD *
* (c) 1998-2007 Petter Nordahl-Hagen. Distributed under GNU GPL v2 *
* DISCLAIMER: THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTIES! *
* THE AUTHOR CAN NOT BE HELD RESPONSIBLE FOR ANY DAMAGE *
* CAUSED BY THE (MIS)USE OF THIS SOFTWARE *
* More info at: http://home.eunet.no/~pnordahl/ntpasswd/ *
* Email : pnordahl@eunet.no *
* CD build date: Mon Apr 9 15:33:39 CEST 2007 *
*****
Press enter to boot, or give linux kernel boot options first if needed.
Some that I have to use once in a while:
boot nousb - to turn off USB if not used and it causes problems
boot irqpoll - if some drivers hang with irq problem messages
boot nodrivers - skip automatic disk driver loading
boot: _
```



<http://www.techrepublic.com/blog/windows-and-office/reset-lost-windows-passwords-with-offline-registry-editor/>

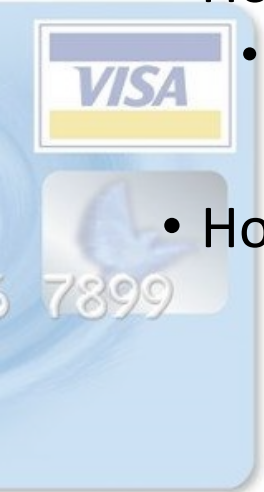
Apple and You?

- In August of 2012, Wired Magazine editor Mat Honan had his Apple account penetrated
- The perpetrators used Mat's Apple account to remotely erase all data on his iPhone, iPad, and MacBook
- This was accomplished by using what multiple companies knew about Mat to put together a complete profile

-Ariel Zambelich, Wired

Who ARE you?

- The perps proved they were Mat, which let them reset Mat's Apple password, and then reset his equipment
- How can you prove you're Mat?
 - Apple says that Mat is the last four digits of his credit card
- How do we get these last four digits?



How to become Mat



1. Call Amazon, tell them you are the Account Holder
 - You'll need Name, Email address, Billing address
2. Add a credit card over the phone
3. Hang Up
4. Call Back, tell them you've lost access to your account
 - You'll need Name, Email address, Billing address, and a credit card number
 - They let you add a new email address: use yours
5. From the web, reset the password, using your new email
6. View the last four digits of the credit cards in the account

Security Isn't Easy

- That's literally all this slide says

