

# Forcing Pandora's Box on Them

Alexander Zangerl

Bond University

az@bond.edu.au

# What is this all about?

- About teaching Unix System Administration *effectively*
- My Tools and Experiences
- A work in progress...
- ...but with some usable tools to show!
- Demonstration of environment used to challenge students

# Who am I?

- Professional Bugbear for students at Bond Uni, QLD
  - see for example  
<http://james.bond.edu.au/courses/inft13333/>
- Sysadmin by choice
- Unix aficionado
- Technician, not priest: not interested in MS voodoo

# Teaching System Administration

- SA is applied and practical, not just pure science
  - Learning by Doing wins big
  - Practice is necessary
- But not everything is safe to play around with
- And the safe things often don't reinforce the crucial skills and mindset
- Problem solving skills are very important!
- Goal: making learning effective but not overly painful...
- ...while still covering the nasties of SA

# I Like Challenges!

- I want students to deal with real disasters
- Without that being a disastrous experience
- But how can a disaster be “benign” without being useless?

# Design Considerations

- Making disaster challenges realistic,
- But still practical and efficiently deployable
- Repeatable, not threatening the student's usual work environment
- Simple invocation required, must not confuse students with Yet Another Complex Concept

# Possible Solutions

## Breaking Things as Needed:

- Provide a tool that “carefully destroys” the environment
- Pretty drastic, fraught with danger, all-or-nothing
- Backing out of partial fixes = ?
- CQU uses this approach

## Virtualisation:

- Simulate the brokenness convincingly, somehow
- Using Virtual Machines, Emulators, Sandboxes
- Also useful for other purposes

# Our Environment

- Small Lab of Linux boxes
- Firewalled off the main Bond Net
- Most machines student-administered
- Small groups of students share a machine
- Central Authentication, but no central file server
- Slow network Hardware
- (but all under my control)

# Our Approach

- We provide pre-broken virtual machines
- Independent instances for every user
- Take care of efficiency of storage and deployment
- Virtual machines to simulate hosts on the lab network, not a full virtual lab on a box

# Toolkit

- Debian Linux as underlying OS variant
- User Mode Linux for virtualisation
- `rootstrap` for bootstrapping a virtual box
- `pandora` glueing all this together flexibly

# What is Debian?

*“The Universal Operating System”*

- a bunch of people with a common goal
- a mindset, expressed by some rules and policies
- an OS software distribution, mostly Linux-centric

*“The Distribution for Sysadmins by Sysadmins”*

# Debian for Teaching

- Consistency of the system aids understanding of principles
- Good automation support
  - For initial installations, in bulk, unattended
  - Also unattended, non-interactive, remote ongoing administration
  - Which scales well - herding a lab of boxes is painless
- Very good flexibility
- Well documented

# User Mode Linux

- Port of Linux kernel to system calls (instead of hardware)
- Provides Virtual OS, not hardware emulation
  - low overhead, efficient and fast
- Runs its own scheduler and VM system
- UML processes and kernel run as processes on host
- Disk storage done via files on host system
- Various levels of networking possible
- No special privileges required!

# Features, Requirements

- Network access to outside network is simple
- Can access host filesystem (or part thereof),
- But works best with separate filesystem images
- Consoles can be mapped onto XTerms, serial lines, pipes...
- Offers Copy-On-Write Layering:
  - Layers a private read-write device over a shared read-only device
  - Writes are done to private device, reads from private or shared
- Minimum needed to get going:
  - UML kernel
  - Root filesystem (image) to boot

# rootstrap

- Tool for building complete Linux filesystem images
- A set of shell scripts that:
  - creates filesystem image
  - installs a base system onto that
  - optionally customises base system somewhat
- Non-interactive operation
- No extra privileges needed
- actually uses UML internally
- Quick, painless, efficient

# Genesis of a Disaster

Make a baseline image set:

- Make filesystem(s)
- Populate root filesystem with base system
- Boot this in UML
- Massage files and software installed

Derive disaster image set:

- UML-boot baseline R/O, use COW layering for R/W
- Mangle this until sufficiently broken
- Merge baseline and COW file into new disaster image set

Distribute this generic disaster image set

- Not customised for individual users yet!
- Victims^Wusers boot UML instance with disaster image set

# pandora

- My tool to automate (most of) the steps
  - Frontend for `rootstrap` (partially completed)
  - Merging of COW files
  - Simple invocation for users to boot the challenge
  - Allows users to reset a challenge to the initial state
- Configurable for different disasters, disk layouts, network environments...
- Extra functionality: Customising disasters on the fly
- Perl, so easily hackable

# Customisation

- Disaster image sets are generic, shared between users on a machine
- But multiple students can run challenges concurrently
  - Use COW files for students' private instances
  - Customise at least network setup to make instances unique per user+machine
- `pandora` takes care of that on first run of challenge
  - Boots a blind+invisible UML with itself as `init` and host filesystem mounted
  - This fake `init` does customisation
  - Then halts and boots the real UML instance with consoles attached to `XTerms`

# Demonstration

- brought six example disasters
- actually used in 2004's run of the subject
- Students were briefly introduced to pandora, the idea of virtual machines and how to terminate a virtual machine the hard way (`skill linux`)
- One needs to know about this much:

```
Usage: pandora [-r|-m newdid] disasterid [extra kernel
parameters...]
-r: reset disaster
-m: merge personal image into new baseimage
```

# Problem 1

User "test" with password "humbug" can log in but has no access to his homedirectory. The user also cannot edit any files with vi.

## Problem 2

None of the main programs are available after boot, not even vi, despite having been installed properly earlier.

## Problem 3

Shortly after starting the system the log daemon stops logging new entries. The system is very busy and logs are being generated by applications all the time, so there must be something seriously wrong.

## Problem 4

This box is shot, badly. It doesn't even boot. But you've got sash installed. The `init=trick` might help a bit. Where's all our programs? Do any binaries work at all in there? If not, what does a dynamically linked binary require to run? That's what I'd test! Eventually you'll be able to boot the thing properly, but make sure to check with `ldconfig -v`. That must work!

## Problem 5

"apt-get update doesn't work. We didn't change anything. It must have broken by itself." Fortunately this is very much a minor foul-up.

## Problem 6

"It doesn't boot. We didn't change anything. It has broken by itself." Fortunately, this time the breakage is relatively minor.

# Observations and Experiences

- Good results with `pandora` environment so far
- Convincing simulation of OS (except low-level hardware stuff: partitions, boot setup)
- One run of the `sysadmin` subject so far, minor problems
- Other subjects use it as general Jail/Sandbox/Test Environment
- `pandora` is being extended for greater flexibility right now
- No substantial problems of understanding for students, just minor confusion occasionally: virtual vs. real vs. remotely accessed machine...

# Problems and Limitations

- Challenges with broken filesystems don't allow for on-the-fly customisation
  - pandora would try to boot these...
  - Worked around this by not enabling networking for such challenges
- Student-administered hosts occasionally lacked configuration for UML
  - Certain kernel features must be enabled for UML to work
  - Newbies got confused by UML's (suboptimal) reporting of such (The challenge doesn't even start, let alone boot...)

# Limitations

- pandora is geared towards running one guest instance at a time
  - Can't do a Virtual Lab on a single computer
  - But there are MLN and other tools for such scenarios
- Baseline images can't be updated without voiding associated COW files
  - Makes a good point for producing standalone disaster image sets...
  - ...instead of layering COWs over COWs over images (which UML doesn't supported so far anyway)

# Positive Experiences

- One Size Fits All (sort of)
  - I use the challenge framework to prepare new challenges
  - Also for other spin-off uses: testing software, Debian packages etc.
  - One baseline image set basis for all this.
  - Needed to configure the baseline system for our environment once only
- Resource Efficiency is good
  - Maximum amount of data is shared between users
  - Images are Sparse Files: only blocks with data are stored
  - Six example disasters together 600MB (while nominally 2.2GB each)

# More Goodies

- Simple to deploy
  - Machine needs: UML, pandora script, filesystem images
  - Filesystem images the same for everybody, easy to distribute
  - But resulting instances can have unique network identity
- Substantial non-challenge uses for experimenting students
  - Provides a playground for doing silly things
  - And the main environment never touched or threatened
  - I actually expected this to be less important to users.

# The Future

- Thinking about automating the breakage procedures
- Make `pandora` a better frontend for `rootstrap` for initial bootstrapping
- More extensive customisation features
- More flexibility for the networking setup

# Questions?

- Feel free to ask!
- ...or later: `az@bond.edu.au`
- You can find `pandora`, the paper and these slides at `http://people.debian.org/~az/pandora/`