

open



USE



IMPROVE



EVANGELIZE

BrandZ : A Better Linux Platform



Vineeth Pillai
Sun Microsystems

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
⋮
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



Agenda

- Overview
- Why BrandZ??
- BrandZ – Zones Integration.
- Implementation
 - Kernel details
 - Interposition points
- The 'Ix' Brand
 - Features
 - Observability
 - Limitations
- Q&A



BrandZ – An Overview

- A Branded Zone is an application environment within the Solaris operating system that enables Binary Applications of another operating system to run unmodified on OpenSolaris.
- It is NOT Xen. It is NOT VMWare
- Currently the 'lx' brand is supported .
- It is the result of Project Janus and Solaris Containers for Linux applications (SCLA)



Why BrandZ ?

- As a conversation starter that gets us in the door
 - “Yeah, I know you're mostly a Linux shop, but let us show you what Solaris can do with your Linux applications”
- As a transition tool, reducing the Linux “barrier to exit”
 - Customer would like to move to Solaris, but has legacy Linux applications
- For developer/ISV capture
 - Solaris could be a better Linux development platform than Linux



BrandZ – Zones Integration

- A Brand is an attribute of a zone, set at zone create time
 - `zonecfg create -t <brand>`
- Each Brand provides a configuration specification, install scripts, boot scripts, and assorted libraries
 - Live in `/usr/lib/brand/<brand>/`
 - Install script load software into the zone
 - boot/halt scripts allow the brand to perform any custom configuration, etc
- `'zoneadm list'` will report brand types that are supported



BrandZ – Kernel Support

- BrandZ infrastructure provides a set of interposition points:
 - syscall path, process loading, signal delivery, etc
- Interpositioning allows a Brand to replace or modify Solaris behavior with its own
- Macro `BRAND_CALLBACK()` placed at the top of all entry point handling routines
 - Checks whether the process is a native process or a branded process



BrandZ – Kernel Support (Contd..)

- Interpositioning only applies to processes in a branded zone (no performance impact on native Solaris processes)
- Fundamentally different brands may require new interposition points
- Only one new brand specific system call is being added to the system: `SYS_brand`



Kernel Integration

- Brands loaded as brand modules

/onnv/onnv-gate/usr/src/uts/common/sys/modctl.h

```
/* For Brand modules */
struct modlbrand {
    struct mod_ops      *brand_modops;
    char                *brand_linkinfo;
    struct brand       *brand_branddef;
}
```



Kernel Integration (contd...)

- Each brand should declare struct brand.

```
/onnv/onnv-gate/usr/src/uts/common/sys/brand.h
```

```
/*  
 * The b_version field must always be the first entry in this  
 * struct.  
 */  
typedef struct brand {  
    int          b_version;  
    char         *b_name;  
    struct brand_ops  *b_ops;  
    struct brand_mach_ops *b_machops;  
} brand_t;
```



The 'Ix' Brand

- SCLA is implemented via the 'Ix' Brand
- The 'Ix' Brand only officially supports RHAS AS v3 and the corresponding CentOS releases. (These distros are based on a Linux 2.4 kernel.)
- 'Ix' brand zone install script populate a zone with Linux bits
 - Customer must provide RHAS or CentOS install media
 - Will also support flash archive-like installation from a tarball



Lx Brand – An Overview

- Creates a zone for Linux application execution
 - Zone is populated only with Linux software
 - Runs Linux init(1M) and configuration scripts
 - Includes basic /proc and /dev support
- There are no Linux binaries or packages delivered with BrandZ
 - This is not a Linux distro and we do not include our own special Linux software
 - We install and run standard Linux distributions
 - Can Install: Red Hat AS version 3 and the equivalent CentOS release



Lx Brand - Installation

- Three options in front of us:
 - Execute distribution's installation tool directly
 - Develop custom installer for a distribution
 - Unpack pre-built archive of installed image.



Lx Brand – Interposition points

- `init(1M)`
- `exec`
- `syscall`
- Threading
- Signal handling
- Devices
- `ioctl`s
- `/proc` filesystem
- Debugging and observability



init(1M)

- Init should be spawned from kernel (SMF requirement)
- kill can't send signals to Linux init.
- Handle runlevels



Loading and running Linux Binaries

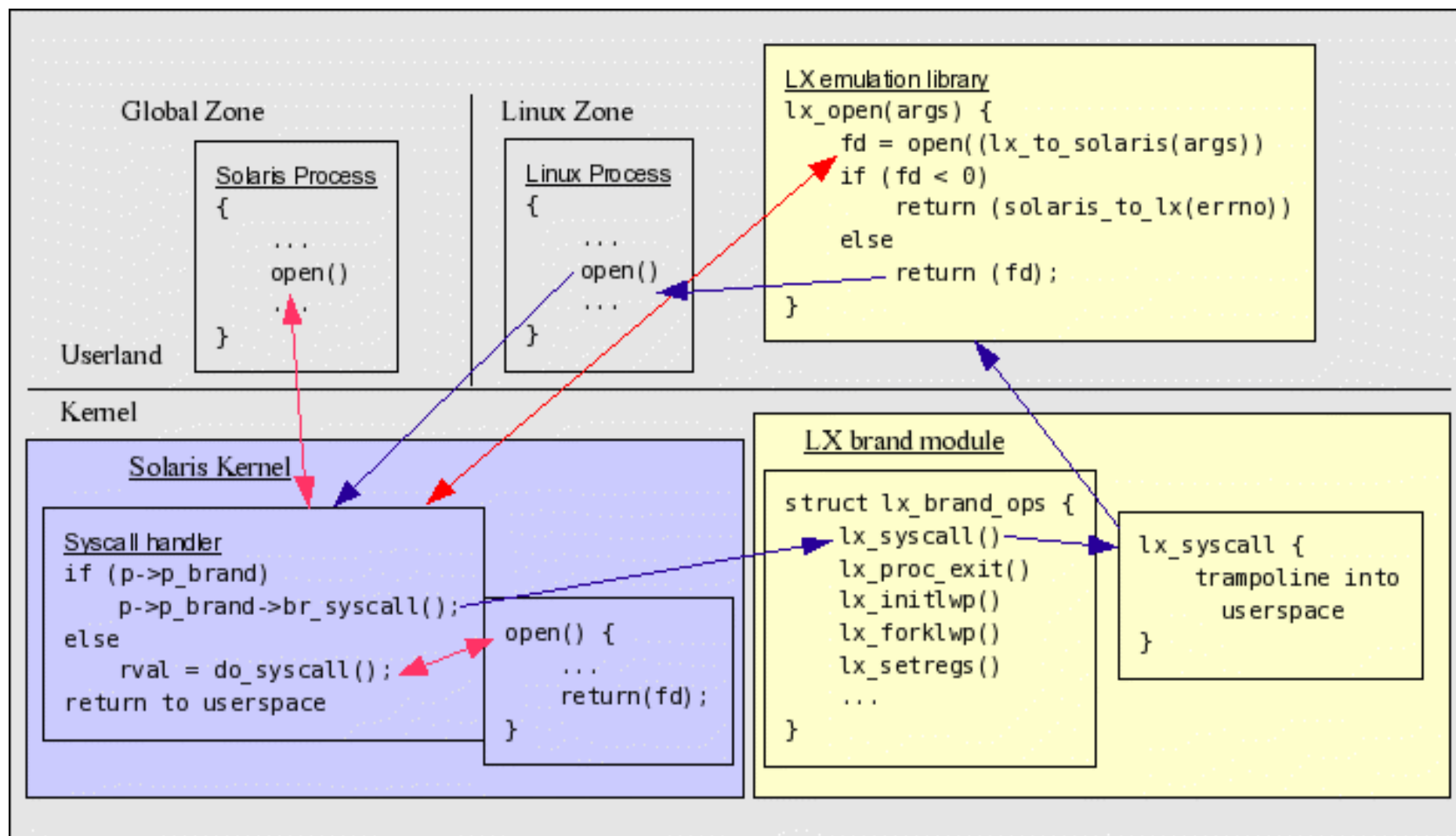
- Brand's exec handler takes over the control.
- Maps non-native ELF and its interpreter into the address space
- Places extra info needed for the interpreter
- Hands over control to brand support library
- Library runs the initialization code
 - Registers to kernel via `brandsys(2)` system call
 - Passes the data structure `struct lx_brand_registration`.
- Jumps to interpreters entry point and execution continues.



System call Emulation

- Need to have a handler for int 80.
- Three types of Emulation
 - Pass through
 - ♦ eg: open, close read, write
 - Simple emulation
 - ♦ eg: stat, mlock, getdents
 - Complex Emulation
 - ♦ eg: futex, clone
- Return Error values may differ.
 - Handled by `lx_emulate`.

Syscall Mapping - Example





Threading

- Linux Threading implementation via clone() syscall.
 - Emulated via fork() and lwp creation.
- Linux Threads have unique pid.
 - Managed by some reserved pids.
- Solaris and Linux use %gs to access per-thread data. So %gs usage is virtualised.



Signal Handling

- Differences
 - Signal numbering and count.
 - Stack structure
 - Signal datastructures.
- Signal Delivery.
 - Need to handle use of %gs register(`lx_sigacthandler()`)
 - Translate solaris signal number to linux and viceversa(`lx_call_user_handler()`)
 - `setsigaction()` to install `lx_sigacthandler()`.



Signal Delivery

```
kernel ->
  sigacthandler() ->
    call_user_handler() ->
      user signal handler
```

for BrandZ Linux threads, this instead would look like this:

```
kernel ->
  lx_sigacthandler() ->
    sigacthandler() ->
      call_user_handler() ->
        lx_call_user_handler() ->
          Linux user signal handler
```



'lx' - Observability

- Support for both Solaris and Linux tools
- In the Linux zone:
 - strace – syscall tracer
 - gdb – GNU debugger
- From the Global zone:
 - Dtrace: will have PID provider, Linux syscall provider, and maybe SDT probes in the Brand library
 - mdb: will be able to manipulate live processes and core files
- Goal: to be a better Linux development platform than Linux



'ix' - Features

- A limited subset of Linux /proc is supported via a new filesystem module
- A limited subset of Linux devices are supported. (mostly virtual devices required to run any applications. ex : /dev/null, /dev/random, pseudo terminals, etc.)
- OSS audio devices are supported
- NFS client mounts and NFS auto-mount are supported



'ix' - Limitations

- Linux applications will not run in the global zone
- On a crash, applications will leave Solaris core files
- Cannot support Linux kernel modules
- Cannot access Linux file systems
- Access to most physical devices is not supported (e.g: disks, USB devices, frame buffers, etc)



References

- brandz-discuss@opensolaris.org
- <http://www.opensolaris.org/os/community/brandz>
- <http://www.opensolaris.org/os/community/brandz/design/>
- <http://www.opensolaris.org/os/community/brandz/design/>

open



USE



IMPROVE



EVANGELIZE

Thats All Folks !

Vineeth Pillai
Sun Microsystems

vineeth.pillai@sun.com

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை