FOSDEM 2012: libvirt-sandbox

Building application sandboxes on top of LXC and KVM with libvirt

Daniel P. Berrangé <berrange@redhat.com>

Talk Overview

- Introduction & Background
- Design & Implementation
- Usage Scenarios

Introduction & Background

OS Access Control

- DAC: isolates user / group identities
  - File 'foo' owned by 'fred' cannot be written by 'bob'
- MAC: isolates processes / objects
  - Process running type 'foo_t' cannot access files labelled 'bar_t'
  - Strong isolation of apps
  - Complex policy
  - Tradeoff between security vs flexibility

Application Sandboxes

- Isolate general purpose applications
- Target specific use cases
- Variety of approaches
  - Seccomp – Linux syscall restriction
  - Java VM – bytecode verification
  - SELinux – MCS isolation
  - Virtualization – OS separation
- Multiple layers of defense

SELinux Sandbox

- Type enforcement
  - Coarse functional areas, min, net, X, web
- Multi-category security
  - Isolation across types via unique categories
- Namespaces
  - Private /tmp & $HOME
  - Copy-in files to private areas
- ...

SELinux Sandbox

- Graphical display
  - XNest
  - Simple WM w/o titlebars
- Resource controls
  - Cgroups
  - Hard to configure

Virtualization

- Server / Cloud
  - Utilization / hardware on demand
- Desktop
  - Legacy application/OS access
- Full machine virt
  - Xen, KVM, VMWare, etc
- Container virt
  - Solaris Zones, OpenVZ, LXC

libvirt

- Standard, simple, secure C API
- API bindings
  - Perl, Python, Java, etc
- Mapping to object models
  - SNMP, GObject, CIM, QMF
- Remote RPC access
  - SSH, TLS, GSSAPI

libvirt

- Virtualization
  - KVM, Xen, VMWare, VBox, Hyper-V, OpenVZ, LXC
- Storage
  - VM, SCSI, iSCSI, FiberChannel, NFS, etc
- Networking
  - Bridge, bond, vlan, filtering, shaping
- Host devices
  - PCI/USB device assignment

libvirt: KVM

- Full machine virtualization
- QEMU+KVM+SeaBios
- Boot BIOS or kernel+initrd
- sVirt: SELinux TE + MCS
- Firewall: ebtables/ip[6]tables
- Host FS passthrough p9fs
- CGroups resource controls

libvirt: LXC

- Container virtualization
- Boot “init” binary
- sVirt SELinux TE + MCS
- Firewall ebtables/ip[6]tables
- Host FS passthrough bind mounts
- CGroups resource control

Design & Implementation

libvirt Sandbox Goals

- LGPLv2+ licensed
- An API for launching sandboxes
- CLI tools for launching sandboxes
- Choice of any libvirt driver
- Choice of any application to confine

libvirt Sandbox Goals

- The 3rd virtualization use case
  - Server, Desktop & **App Library**
- Run applications from the host
  - Not JEOS (Just Enough Operating System)
  - Try NAOS (Not Any Operating System)

libvirt Sandbox API

- Based on GObject object system
- Uses libvirt-{glib,gconfig,gobject}
- Accessible from non-C via introspection
- All CLI tools built on top of the API

libvirt Sandbox “Hello World”

```
#!/usr/bin/gjs

const LibvirtGObject = imports.gi.LibvirtGObject;
const LibvirtSandbox = imports.gi.LibvirtSandbox;
const Gtk = imports.gi.Gtk;

LibvirtGObject.init_object_check(null, null);
...
```

libvirt Sandbox “Hello World”

```
var cfg = LibvirtSandbox.Config.new("sandbox");
cfg.set_command(["/bin/ls", "-l", "/", "--color"])
cfg.set_tty(true);

var conn = LibvirtGObject.Connection.new("qemu:///session");
conn.open(null)

var ctxt = LibvirtSandbox.Context.new(conn, cfg);
ctxt.start();
...
```

libvirt Sandbox “Hello World”

```
var console = ctxt.get_console()
var closed = function(error) { Gtk.main_quit(); }
console.connect("closed", closed);
console.attach_stdio()

Gtk.main()

console.detach()
ctxt.stop();
```

virt-sandbox command

- Simple invocation (R/O root, no network)
  - virt-sandbox [OPTIONS] BINARY [ARGS...]
- Choose virt driver using option
  - --connect LIBVRT-URI (or -c LIBVIRT-URI)
- Run 'date' inside LXC
  - virt-sandbox -c lxc:/// /bin/date
- Run 'cat /proc/cpuinfo' inside KVM
  - virt-sandbox -c qemu:///session /bin/cat /proc/cpuinfo

virt-sandbox command

- Bind host files/dirs to guest R/W
  - --host-bind GUEST-PATH=HOST-PATH
- Create empty /home/fred with tmp file
  - --host-bind /home/fred=
- Create /home/fred from /tmp/home
  - --host-bind /home/fred=/tmp/home
- Create /home/fred from /tmp/home.img
  - --host-image /home/freq=/tmp/home.img

virt-sandbox command

- Bind /etc/krb5.conf from /tmp/krb5.conf
  - --guest-bind /etc/krb5.conf=/tmp/krb5.conf
- Copy /home/fred/.firefox into guest
  - --include /home/fred/.firefox
- sandbox.img w/ firefox prof & krb5 conf
  - --host-image /tmp=/home/fred/sandbox.img
  - --guest-bind /etc/krb5.conf=/tmp/krb5.conf
  - --guest-bind /home/fred=/tmp/home
  - --include /home/fred/.firefox

virt-sandbox command

- Add DHCP configured NIC
  - --network dhcp
- Add static configured NIC
  - --network address=192.168.1.1/255.255.255.0
- SELinux dynamic config
  - --security label=svirt_sandbox_t,dynamic
- SELinux static config
  - --security label=svirt_sandbox_t:s0:c123,c123;static

KVM startup

- Boot from host kernel image
- Build initrd w/ hand picked modules+init
  - virtio-9p, virtio-block & virtio-net
- /usr/libexec/libvirt-sandbox-init-qemu
  - Loads kmods
  - Mounts 9p filesystems
  - Mounts block images
  - Mounts special filesystems (/dev, /proc, etc)
  - Creates device nodes

LXC startup

- Boot custom init binary
- /usr/libexec/libvirt-sandbox-init-lxc
  - Reads args from $LIBVIRT_LXC_CMDLINE

Common Startup

- /usr/libexec/libvirt-sandbox-init-common
  - Starts admin debug shell
  - Configures network interfaces
  - Sets up guest bind mounts
  - Drops privileges / capabilities
  - Runs application command
  - Handles I/O forwarding & escaping
- Exits when app closes primary console

Performance Overheads

- Overheads:
  - Fixed startup penalty
  - Ongoing CPU execution penalty
  - Filesystem / network access penalty
  - Fixed shutdown penalty
- Consider relative to overall running time

LXC Performance Overheads

- Startup penalty
  - libvirt container start (<200 ms)
  - libvirt-sandbox init (~0 ms)
  - Total: < 200ms
- CPU execution penalty: nil
- Device access penalty: nil/negligable
- Shutdown penalty
  - libvirt container stop (< 100ms)

KVM Performance Overheads

- Startup penalty
  - Initrd creation (~300 ms)
  - SeaBios startup (~20 ms)
  - Kernel/initrd option ROM (~1000 ms)
  - Linux boot (~100 ms)
  - libvirt QEMU start (<400 ms)
  - libvirt-sandbox init (~20 ms)
  - Total: ~ 3000 ms (3 secs)

KVM Performance Overheads

- CPU execution penalty: near native
- Device access penalty: 90+% of native
- Shutdown penalty
  - Linux poweroff (~50ms – was 1050ms)
  - Libvirt guest shutdown (~200 ms)

Usage Scenarios

Example: Server Virtual Hosting

- Goal:
  - Deploy multiple Apache virtual hosts
  - Strong isolation between virtual hosts
- Solution:
  - One apache instance per virtual host
  - Run apache inside a sandbox

virt-sandbox-service

- virt-sandbox-service create /usr/sbin/httpd
  - Config /etc/libvirt-sandbox/httpd-foo.cfg
  - SystemD unit /etc/systemd/system/httpd-foo.service
  - Create state directories or image
  - Allocate unique MCS security label

virt-sandbox-service

- virt-sandbox-service start httpd-foo
  - Starts service from config
  - Invoked by systemd unit
- virt-sandbox-service stop httpd-foo
  - Invoked by systemd unit
- virt-sandbox-service console httpd-foo
  - Access to admin debug shell

Example: Audio Transcode

- Obtained 'ogg' from untrusted source
- Decode to 'raw' format
- Prevent all filesytem & network access
- Only R/W on stdin/out
  - ```
    virt-sandbox -c lxc:/// -- /usr/bin/oggdec -o - - \
      < /path/to/untrusted.ogg \
      > /path/to/trusted.raw
    ```
    ```

Example: mock RPM Build

- setgid binary for users in 'mock' group
- Installs chroot with target distro RPMs
- Runs RPM as 'mock' user inside chroot
- Problem:
  - RPM chroot install runs as 'root'
  - RPM %post/%pre scripts run as 'root'
  - 'root' user can escape any chroot
    - => Malicious %post/%pre scripts can escape chroot

Example: mock RPM Build

- Solution:
  - Install chroot using 'rpm' in a sandbox
  - %pre/%post scripts run as 'root'
  - 'root' cannot escape from sandbox
    - => %pre/%post scripts cannot escape

Example: Web Browser

- Problem:
  - Differing security requirements
    - “Social Networking” vs “Online Bank”
  - One shared cookie store
  - One shared password database
  - One shared plugin config
  - No DAC or MAC isolation between sites
    - => Compromise of browser is high impact

Example: Web Browser

- Run browser instances in sandbox
- Separate instances per security needs
  - One for “general” use (social networking)
  - One per online banking/financial site
  - One for/per online shopping

Example: Web Browser

- Separate $HOME/.firefox per instance
  - Individual cookie jar
  - Individual password / form data store
  - Individual certificate store
    - => Protected by MAC
- Pre-populate cert store for banking sites
  - => Avoid certificate spoofing

Example: Web Browser

- Interaction via SPICE/VNC
  - Browser cannot attack desktop apps
  - Browser cannot attack audio service
  - Unique decorations for high security sites
- Network filtering via ebtables/iptables
  - Block untrusted sites connecting to trusted sites

FOSDEM 2012: libvirt-sandbox

http://libvirt.org/