Collabora

LibreOffice
The Document Foundation

SUSE

Faster Jail Creation with Bind-Mount

Collabora Office

OPENSUSE-LIBREOFFICE CONF'20

2020

By Ashod Nakashian
Consultant at Collabora Office
ash@collabora.com

Talking Points

Part I: Overview and Background

- Chroot and Sandboxing
- What are SysTemplate and LoTemplate?
- The Naive Approach

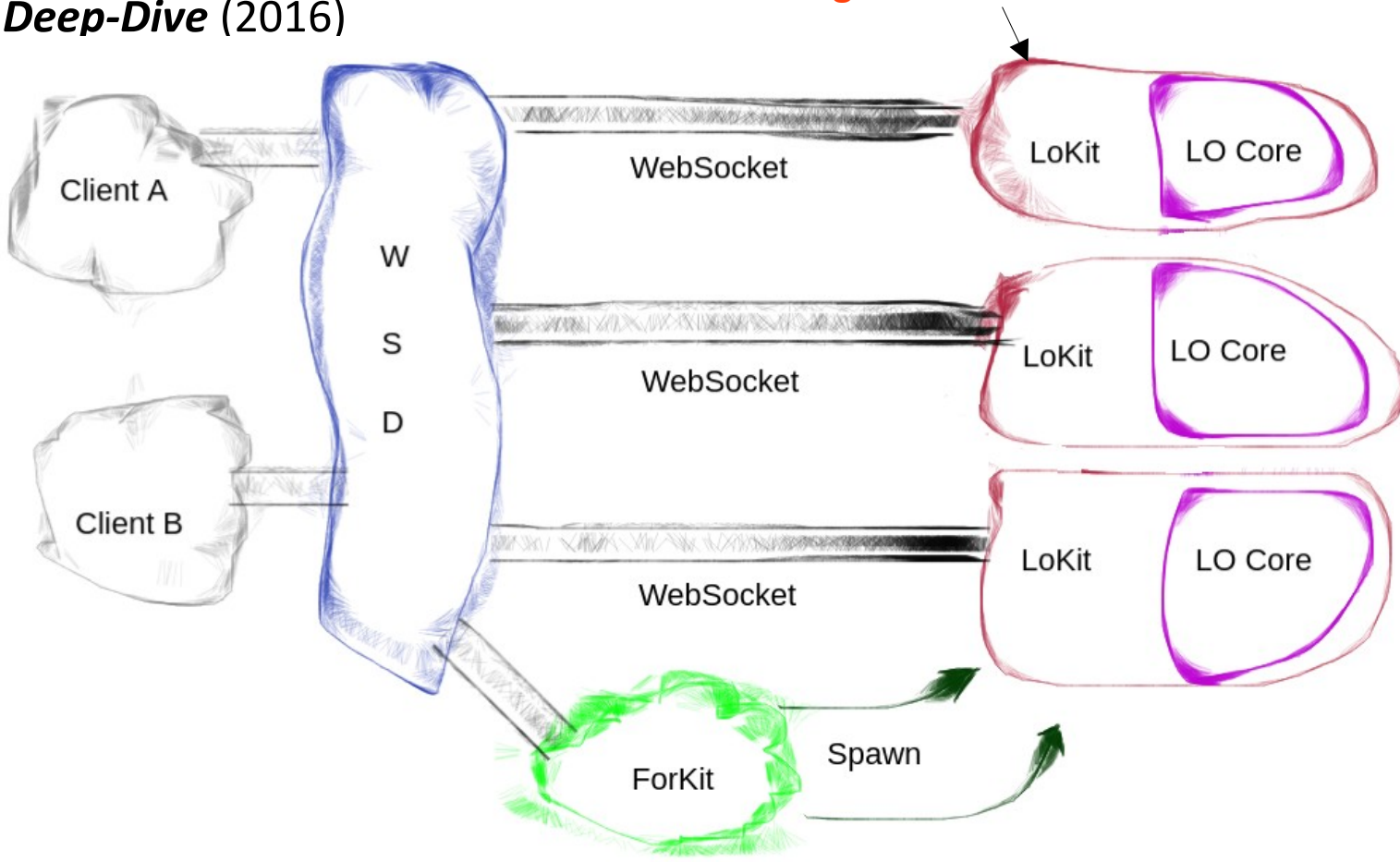
Part II: Bind-Mount

- Overview
- Challenges
- The New Strategy

Part I
Overview and Background

Architecture

See my talk ***Online: Deep-Dive*** (2016)



Chroot and Sandboxing

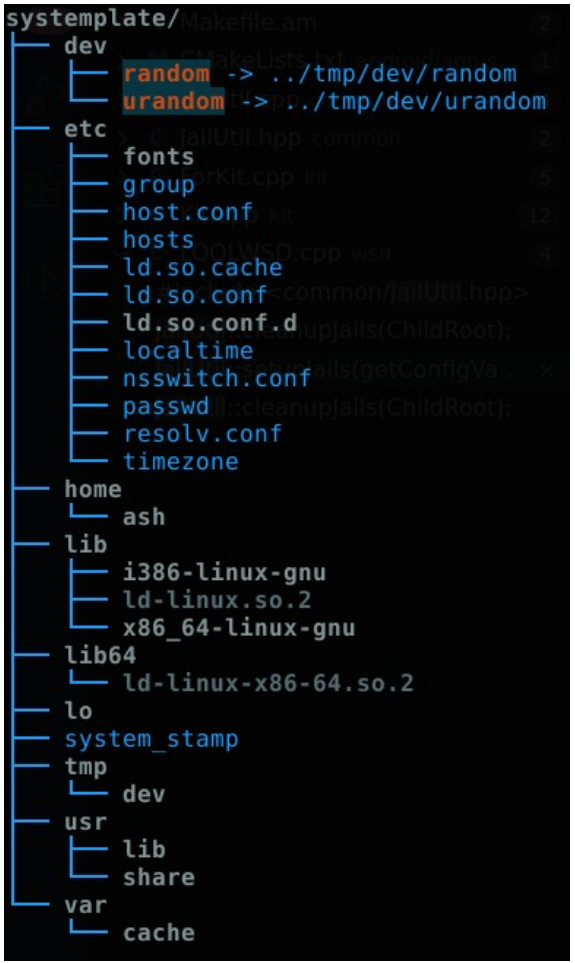
Process Isolation

- Each document is loaded in a dedicated process (LoKit, or Kit for short)
- The filesystem is isolated via `chroot(2)`
- Once chroot is called, we drop privileged capabilities, including `CAP_SYS_CHROOT` and `CAP_MKNOD`

SysTemplate and LoTemplate

The Jail “master” template

- To create the jails, we prepare a template called `SysTemplate`
- The `SysTemplate` has all the required files, including system libraries
- SysTemplate is created via `loolwsd-systemplate-setup` script at installation
  - But some files need regular updating (more later)
- There are typically thousands of files and hundreds of megabytes of data in the `SysTemplate`
- And the LibreOffice installation: `LoTemplate`



Jail Bootstrapping

The naive approach

- Jail directories are created in a configurable root directory
- Each jail root directory is given a cryptographically-secure random name
- *`SysTemplate` content files are linked into the jail directory*
  - *If linking fails, the files are copied*

Jail Bootstrapping (continued)

Limitation of the naive approach

- Linking several *thousand* files is fast only on SSD drives and outside of containers
  - Production hardware can expect to link all files in under ~200ms
- Copying, however, is painfully slow on anything but the fastest SSDs
  - Even then, copying is at least an order of magnitude slower than linking
- Inside containers, such as Docker, the performance of linking can be as bad as 40+ ms per link
  - Meaning, each 1’000 files will take several *seconds*
  - Slow enough that loading document can timeout and fail
- Less critical: cleaning up still has to deal with the 1’000s of files in the jail

Part II
Bind-Mount

Bind-Mount

Overview

- Mounts a directory-tree at the given path
  - Unlike disk or file-system mounting, bind-mounting only supports existing paths
  - Support for read-only mounting, recursive, and many options
- `mount(2)` allows us to mount a complete directory with a single syscall
- In theory, we shouldn't need more than one mount call per jail (i.e. per doc)
- `umount2(2)` allows us to unmount, for an equally fast clean up
- As `mount(2)` and `umount2(2)` need `CAP_SYS_ADMIN`, managing mounts is done via a dedicated process that has the necessary capabilities: `loolmount`
  - This limits the processes that have elevated privileges, reducing attack vector footprint

Bind-Mount: The Motions

It takes three to tango

- Unfortunately, we can't mount with a single syscall to mount(2)
  - When bind-mounting, we can't also set the read-only flag
- First, we bind-mount
  - `MS_MGC_VAL | MS_BIND | MS_REC`
- Next we make it read-only:
  - `MS_BIND | MS_REC | MS_REMOUNT | MS_NOATIME | MS_NODEV | MS_NOSUID | MS_RDONLY | MS_SILENT`
- Finally, we need to disable re-binding, lest out sub-mounts show up in other jails
  - `MS_UNBINDABLE | MS_REC`

Challenges

Unfortunately, with power comes problems...

- Since a mounted directory shares the source (i.e. `SysTemplate`), it must be read-only
  - Otherwise, a rogue document can modify `SysTemplate`, compromising the server
- We can mount with MS_RDONLY flag to make the jail read-only
  - But, we still need a writeable /tmp and /home directories!
  - Worse, we need to update certain files regularly: `/etc/hosts`, `/etc/resolv.conf`, etc.
    - We could update them in `SysTemplate`, but… (more later)
- Mounting may fail, or indeed be disabled by admins via config, and we must fallback
  - The clean-up method now becomes ambiguous: do we unmount or '`rm -rf`'?

New Strategy to Jailing

A multi-layered approach

- Do as much preparation as possible in `loolwsd-systemplate-setup`
  - Set up the random devices as relative symbolic links: ../tmp/dev/random
- Split the jail management into three parts:
  - `Loolwsd` does the initial setup and ultimately enables the fallback (link/copy) if/when mounting is not enabled or possible
  - `Forkit` updates SysTemplate, but only if it's writeable, also does clean up
  - `Kit` is responsible for the heavy-lifting...

New Strategy to Jailing: The Setup

Inside the Kit, if mounting is available...

- First, mount `SysTemplate`, and make it read-only
- Next, mount `LoTemplate`, and make it read-only
- Create a cryptographically random directory in root directory of jails
  - Bind-mount as `/tmp` in the jail => **not** read-only
- If any step fails, fallback to linking, which falls back to copying
- When the above is done, create the random devices in `/tmp/dev/`
- Setup TMP and HOME environment variables
- Ultimately, 3 logical mounts, each costing 3 syscalls

Read-Only SysTemplate + Other Special Cases

But wait, there is more!

- For added security `SysTemplate` may be owned by root
- This makes it read-only, and can't be updated post installation
- This implies that the dynamic files (`/etc/hosts`, `/etc/resolv.conf, etc`) must be either links (to remain up-to-date), or we must disable mounting and link/copy when they are outdated
- In AppImage and Mobile, SysTemplate is handled in a special way altogether
  - There is no chroot in AppImage, for example
- Many more corner-cases and special cases, either generalized or handled individually
- Much faster performance in both the best and worst case, on the order of milliseconds!

Collabora Office

<Your Question Here>

By Ashod Nakashian

ash@collabora.com