Application sandboxing with xdg-app

David King <amigadave@amigadave.com>

8th May 2015 / GNOME.Asia / Conference Room #4

http://amigadave.com/presentations/gnome_asia_xdg-app_2015.pdf

Goals

- Reduce attack surface available to applications
- Distribute an application for multiple distributions

Technologies

- Namespaces
- cgroups
- SELinux
- kdbus
- Wayland

Problems

- Restricting application access requires changes to applications
- New sandboxed APIs needed
- Dependencies on new (and unfinished) technologies, such as Wayland and kdbus

Concepts

- Runtimes: well-defined set of dependencies bundled together
- App bundles: an application's files, together with some additional metadata
- SDKs: a runtime with only the development headers and build tools

Tools

- ostree: the technology behind applications and runtimes
- xdg-app: general entry point for installing, updating and running applications and runtimes (and everything involved in building them)

Runtimes

- Roughly equivalent to /usr on a traditional system
- Can be installed system-wide or per-user
- Stacking of runtimes is expected
- GNOME is distributing preview runtimes and SDKs, hopefully will be the way that GNOME is distributed in the future
- Preview KDE runtimes available
- Can be signed with GPG (just like ostree)

Filesystem layout of app bundles

- On-disk layout of an app bundle is the reverse DNS name, followed by the architecture and then a branch name. For example, `org.gnome.gedit/x86_64/3.16`
- The `active` subdirectory inside the branch contains the data that makes up the application, in the `files` and `export` subdirectories
- The `files` subdirectory is mounted with a prefix of `/self` inside the sandbox
- Exported files (desktop file, D-Bus service file, etc.) are symbolically linked to a separate `exports` directory inside the user or system configuration path during installation

Filesystem layout of runtimes

- Similar to app bundles (reverse DNS name, `active` and `files` subdirectories)
- Mounted to `/usr`

Filesystem layout of the sandbox

- Each application has its own sandboxed XDG directories (`XDG_DATA_HOME`, `XDG_CONFIG_HOME` and `XDG_CACHE_HOME`)
- XDG directories are preserved across application upgrades

Application metadata

- Desktop file (INI-file/keyfile) syntax
- Bundled with the app to declare the features required from the sandbox
- Any optional feature not explicitly requested will not be enabled
- Specifies required runtime, and build-time SDK
- Also possible to override and unset environment variables (useful for plugins or other extensions)

Application metadata example

```
[Application]
name=org.gnome.gedit
runtime=org.gnome.Platform/x86_64/3.16
runtime=org.gnome.Sdk/x86_64/3.16
sdk=gedit

[Environment]
wayland=true
ipc=true
network=true
session-dbus=true

[Vars]
GEDIT_FOO=bar
```

Runtime metadata

- Similar to application metadata
- Extension point directories and subdirectories can be specified

Runtime metadata example

```
[Runtime]
runtime=org.gnome.Platform/x86_64/3.16
runtime=org.gnome.Sdk/x86_64/3.16
sdk=gedit

[Extension org.gnome.Platform.Timezones]
directory=share/zoneinfo

[Extension org.gnome.Platform.Locale]
directory=share/gnome-sdk/locales
subdirectories=true

[Vars]
GTK_PLUGIN_PATH=/self/lib/gtk/plugins:/usr/lib/gtk/plugins
```

The sandbox

- Uses read-only bind mounts for the runtime and app bundle directories, as well as any extension point directories
- Uses writable bind mounts for the data directories, home and host filesystems (if requested, otherwise `/var/home` is used instead)

Requirements

- OSTree requires hardlinks
- Kernel namespaces
- `clone()` syscall
- bind mounts
- any requested features depend on the feature being present on the host (Wayland, kdbus, PulseAudio)

Pre-requisites

- Runtime and SDK appropriate for the application
- Build system that allows installation into a custom prefix
- Additional dependencies must be built and installed into the app bundle

Populate build directory with metadata

```
mkdir c-gnome-app-build

xdg-app build-init c-gnome-app-build \
        org.gnome.Sdk org.gnome.Platform 3.16
```

Configure and build application

```
cd c-gnome-app

xdg-app build ../c-gnome-app-build ./configure --prefix=/self
xdg-app build ../c-gnome-app-build make
xdg-app build ../c-gnome-app-build make install
```

Collect the bundle contents together

```
xdg-app build-finish --command=c-gnome-app --allow=x11 \
        --allow=session-dbus --allow=network \
        ../c-gnome-app-build
xdg-app build-export ../repos/c-gnome-app \
        ../c-gnome-app-build org.gnome.CHello
```

Install the bundle

```
xdg-app --user add-remote --no-gpg-verify c-gnome-app \
        file:///home/david/checkout/repos/c-gnome-app
xdg-app --user install-app c-gnome-app org.gnome.CHello
xdg-app --user run org.gnome.CHello
```

Explore the app bundle sandbox

```
xdg-app run --devel --command=bash org.gnome.CHello
```

Restrictive access control

- Sandboxed applications have restricted access to the filesystem
- New mediation API is required to provide revokable access
- General term for interaction is “portals”
- Composed of a backend daemon and a UI for permission granting/revokation

gvfs portal backend

- Preliminary support for `document://` URIs via xdg-document-portal
- Basic file access is functional, still needs support for metadata
- gvfs branch: https://git.gnome.org/browse/gvfs/log/?h=wip/alexl/documents
- Requires very small amounts of application porting

Future portals

- Audio
- Webcams
- Printing
- Generic device authorization (Bluetooth, cellular modem, etc.)
- Content sharing
- GSettings/dconf?

Further resources

- Wiki: https://wiki.gnome.org/Projects/SandboxedApps
- xdg-app: https://github.com/alexlarsson/xdg-app
- xdg-document-portal: https://github.com/alexlarsson/xdg-document-portal
- Ubuntu PPA: https://launchpad.net/ amigadave/+archive/ubuntu/xdg-app
- Fedora COPR: http://copr.fedoraproject.org/coprs/amigadave/xdg-app/

Building app bundles with RPMs

- Good way to bootstrap app bundles with extra dependencies
- Relies on a modified `rpmbuild` which builds into `/self`
- Additionally, uses a special “var runtime” to provide a writable RPM database corresponding to the SDK contents