Flatpak Notes

@TechScribe

LICENSING

Table of Contents

CHAPTER 1

WHAT IS FLATPAK?

Flatpak is a desktop application distribution containerization technology, essentially it jails your application into it's own little world through the security feature called Sandboxing.

Let's hypothetically say we have an editor software downloaded and installed on our machine. On it's own it is fairly harmless, but if we add random extension from the marketplace that it may not have been audited by other developers, we are subjecting ourselves to increased risks of running malicious software and codes. It could for instance overwrite our .bashrc script which overrides default bash behavior to alias existing program such as sudo which can then be used to harvest/load malicious software to hijack our machine with ease.

You can install flatpak by running the following for your respective operating system with the handy table below or to refer to this webpage: https://flatpak.org/setup/.

Distribution	Command
ArchLinux	pacman -S flatpak
Debian/Ubuntu/PopOS	apt install flatpak
OpenSUSE/SUSE	zypper install flatpak
Red Hat Enterprise Linux/CentOS	yum install flatpak
Fedora	Already installed
NixOS	Please refer to https://flatpak.org/setup/NixOS for instructions.

You may want to enable flathub repository for installing flatpak application.

```
flatpak remote-add --if-not-exists flathub \
https://flathub.org/repo/flathub.flatpakrepo
```

FLATSEAL TOOL

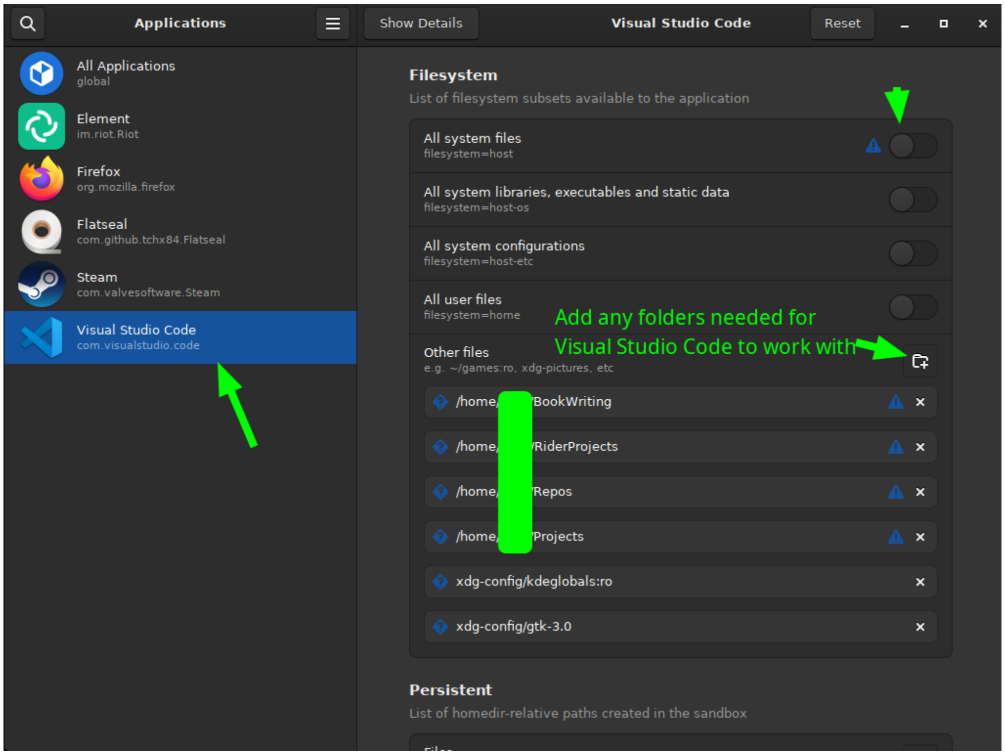
Flatseal Program is a frontend software that allows us to modify the permission of installed flatpak applications and we can limit or restrict what program is permitted to use. We can install flatseal by running the following command:

```
flatpak install flatseal
```

For this demonstration, we will install ”Visual Studio Code” flatpak application and then open it with flatseal to view it’s privilleges.

```
flatpak install com.visualstudio.code
flatpak run com.github.tchx84.Flatseal
```

The privillege ”All system files” permission can be turned off after clicking on Visual Studio Code tab on the left side and you can selectively add specific folders that you want to allow Visual Studio Code to have control over:



All Applications
global

Element
im.riot.Riot

Firefox
org.mozilla.firefox

Flatseal
com.github.tchx84.Flatseal

Steam
com.valvesoftware.Steam

Visual Studio Code
com.visualstudio.code

Filesystem
List of filesystem subsets available to the application

All system files
filesystem=host

All system libraries, executables and static data
filesystem=host-os

All system configurations
filesystem=host-etc

All user files
filesystem=home

Add any folders needed for Visual Studio Code to work with

Other files
e.g. ~/games:ro, xdg-pictures, etc

/home/[redacted]/BookWriting
/home/[redacted]/RiderProjects
/home/[redacted]/Repos
/home/[redacted]/Projects
xdg-config/kdeglobals:ro
xdg-config/gtk-3.0

Persistent
List of homedir-relative paths created in the sandbox

PRACTICAL DEMONSTRATION

Let's suppose we have the following malicious code that it's sole intention is to alias sudo command to run a bad software in substitute.

```
Console.WriteLine("Now attempting to overwrite .bashrc!");
var currentDir = Environment.CurrentDirectory;
var nameOfDir = "";
while (!string.IsNullOrWhiteSpace(currentDir) &&
    currentDir != (Directory.GetParent(currentDir)?.FullName ?? string.Empty)
    && nameOfDir != Environment.UserName.ToLower())
    {
        currentDir = Directory.GetParent(currentDir)?.FullName ?? string.Empty;
        var idx = currentDir.LastIndexOf(Path.DirectorySeparatorChar);
        idx = idx < 0 ? 0 : idx + 1;
        nameOfDir = currentDir.Substring(idx);
    }
var bashrcFilePath = Path.Join(currentDir, ".bashrc");
if (File.Exists(bashrcFilePath))
    File.AppendAllText(bashrcFilePath, "\nalias sudo = badsoftware\n");
else
    File.WriteAllText(bashrcFilePath, "\nalias sudo = badsoftware\n");
Console.WriteLine($"Successfully written bad bashrc file to {bashrcFilePath}");
```

Now we can go ahead and run the program by opening integrated terminal by clicking on Terminal drop down menu in Visual Studio Code and then click on New Terminal menu item and then enter the command into the terminal at the bottom as followed:

```
dotnet run
```

Now you'll find the .bashrc file created in your flatpak home directory after running the following command:

```
nano ~/.bashrc
```

You'll notice that it only contains the malicious code, but nothing else when you open .bashrc. That .bashrc will only affects whatever is **INSIDE** that flatpak application and you won't be affected by it anytime you open your terminal program outside of flatpak package and if you open .bashrc file, you'll notice that it have yet to be modified by the malicious demonstration code above.

That's the power of Flatpak!