Linux Plumbers Conference 2022

>> Dublin, Ireland / September 12-14, 2022

# FUSE-BPF

## Daniel Rosenberg

Software Engineer

## Paul Lawrence

Software Engineer

## Android and Fuse - a checkered past

Many potential use cases for fuse in Android e.g.

- Sdcard emulation
- Incremental file system
- Location Redaction
- Folder hiding

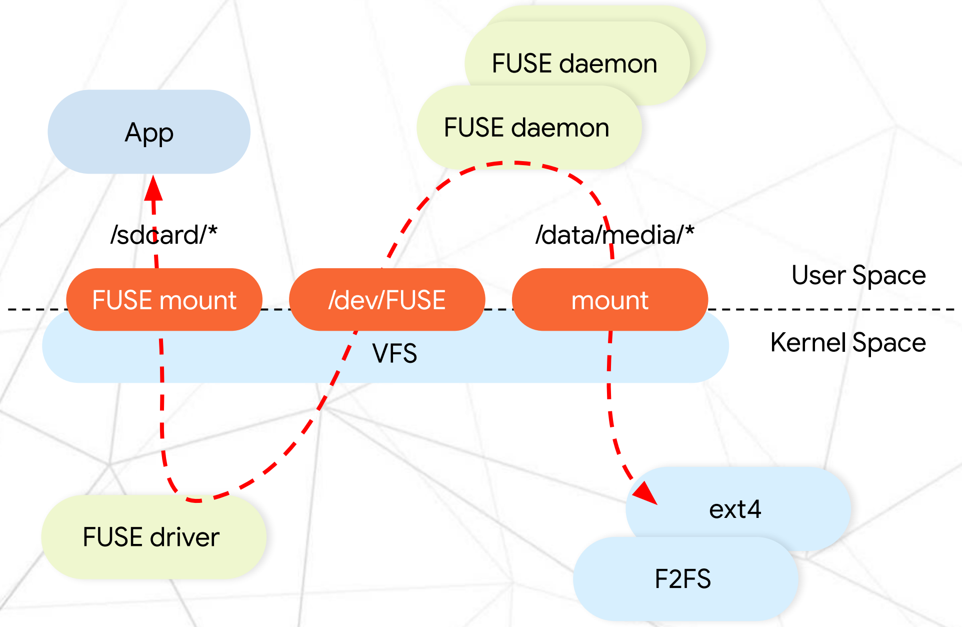
BUT

Android ecosystem requires essentially 0% degradation in latency and throughput, so only actual shipping use case is location redaction

Linux Plumbers Conference 2022

>> Dublin, Ireland / September 12-14, 2022

# FUSE in Android



Extra permission checks on shared storage access, e.g., only some apps can access some folders

Data redaction, e.g., remove metadata from pictures

Emulates the external storage regardless its location, e.g., (un)mounting external storage media

Linux Plumbers Conference 2022
>> Dublin, Ireland / September 12-14, 2022

## Hope - Passthrough and BPF

- Passthrough mode in fuse: allows a fuse daemon to pass all reads and writes on a specific file through to a backing fs. Does nothing to optimize directory operations and open/closes, but reads and writes end up at native speed
- Extfuse: 3 years ago at Plumbers we listened to the talk on extfuse by Ashish Bijlani and Umakishore Ramachandran

We also had requests for directory passthrough, which seemed interesting, but insufficiently flexible.

Singly, none of these ideas seemed likely to resolve our problems, but by combining them we felt we could see a way to use Fuse

## Fuse BPF design

- Generic stacking file system
- Allows requests to be handled by FUSE or by the backing filesystem.
- BPF scripts can pre or post filter each request.
- Both pre and post filters can pass the request on to the FUSE daemon.

# FUSE BPF at a glance

Add to fuse_inode:

- `struct inode *backing_inode;`
- `struct bpf_prog *bpf;`

These may be set at mount time for root, at lookup time for all other inodes

Add a new structure call fuse_bpf_args, which is an analog of fuse_args with only minor differences.

If backing_inode exists, **all** requests will be conditionally sent **to the backing inode**, **else** we are in **classic FUSE** mode

If **no bpf**: simply sent to the backing inode as is (pure **passthrough** mode)

If **bpf**: format `fuse_bpf_args with fuse_bpfin_args` and send to BPF, which may redirect request to **classic FUSE or**

1. Optionally request user-mode pre-filter with same modifiable in_args
2. (Potentially modified) request is sent to backing file system
3. Optionally pass `in_args &out_args` to BPF post-filter
4. Optionally pass `in_args &out_args` to user-mode post-filter

## Use case 1: Hiding external storage app directories

- Problem - we want to hide the existence of certain sub folders from apps, even though those apps have the right to traverse the directory.

- FUSE Bpf solution:
  - Postfilter lookup and convert EPERM to ENOENT
  - Postfiler readdir to remove the to-be-hidden entries.
  - Remove the bpf from the child folders, ensuring native speed at that point.

## Use case 2: Redaction

- Problem: If an app has media file permission but does not have location permission, it can get location data from photos as they are taken.
- Solution: Redact the part of the photo header that contains the location from such apps.
  - Hook opens to insert upper limit of redacted information (use bpf_inode_map)
  - Hook reads to forward reads below this limit to userspace, reads above this limit stay in the kernel

- Note that the current solution redirects all reads and writes to such files from such apps to user space, with significant performance overhead.

## Lookup

Add additional optional return argument:

```
#define FUSE_ACTION_KEEP         0
#define FUSE_ACTION_REMOVE       1
#define FUSE_ACTION_REPLACE      2

struct fuse_entry_bpf_out {
      uint64_t   backing_action;
      uint64_t   backing_fd;
      uint64_t   bpf_action;
      uint64_t   bpf_fd;
};
```

Also support return arguments on negative lookups

Linux Plumbers Conference 2022
>> Dublin, Ireland / September 12-14, 2022

## Verification

- Extended bpf packets
  - Multiple packets per structure to handle varied number of args of varied length
- Helper function for write access
  - Communicates with Fuse via set of flags
  - Converts PTR_TO_PACKET to PTR_TO_ALLOC_MEM_OR_NULL for verification purposes
- Flags set by FUSE

| | | |
|---|---|---|
| BPF_FUSE_IMMUTABLE | - | May not be written to |
| BPF_FUSE_VARIABLE_SIZE | - | May change sizes |
| BPF_FUSE_MUST_ALLOCATE | - | Must be re allocated before writes |

- Flags set by helper function

| | | |
|---|---|---|
| BPF_FUSE_MODIFIED | - | Buffer write was requested |
| BPF_FUSE_ALLOCATED | - | New buffer allocated |

## Remaining work

- Fully support read_iter and write_iter
- Upstream upstream upstream …

## Thank you!

Any questions?