A Year of Container Kernel work

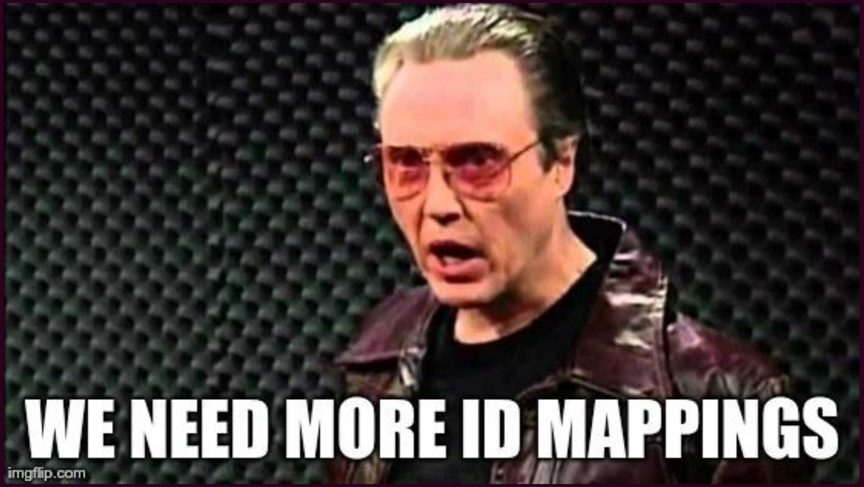
Past, Present, and Future of Container Kernel Features

FOSDEM, 2019
Brussels, Belgium

Christian Brauner
christian@brauner.io
christian.brauner@ubuntu.com
@brau_ner
https://brauner.io/

4.15: Bump Limit of Allowed User Namespace Mappings from 5 to 340

- aa4bf44dc851c6bdd4f7b61b5f2c56c84dfe2ff0
- 6397fac4915ab3002dc15aae751455da1a852f25
- 11a8b9270e16e36d5fb607ba4b60db2958b7c625
- 3edf652fa16562fb57a5a4b996ba72e2d7cdc38b
- d5e7b3c5f51fc6d34e12b6d87bfd30ab277c4625
- ece66133979b211324cc6aff9285889b425243d2
- 3fda0e737e906ce73220b20c27e7f792d0aac6a8



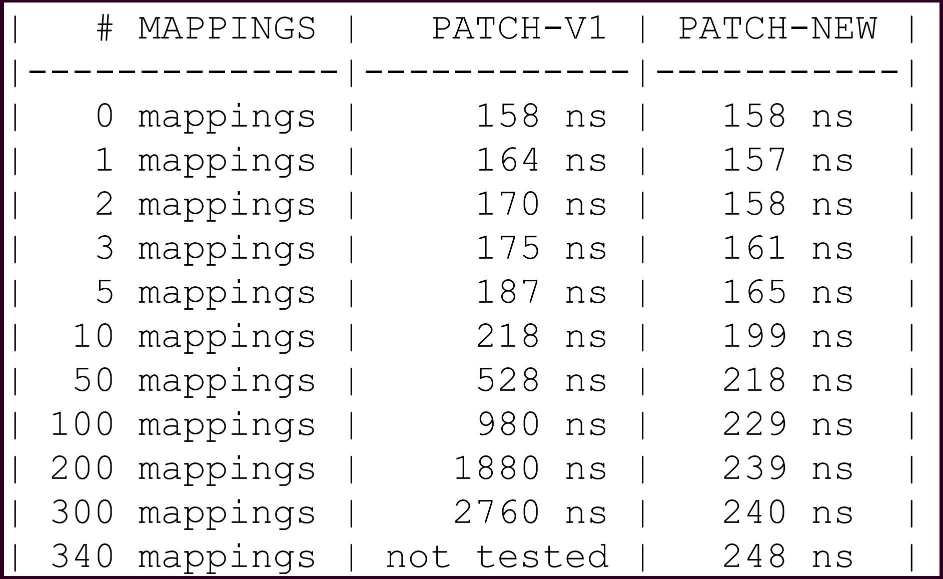
4.15: Bump Limit of Allowed User Namespace Mappings from 5 to 340

```
#define UID_GID_MAP_MAX_EXTENTS 5

struct uid_gid_extent {
        u32 first;
        u32 lower_first;
        u32 count;
};

struct uid_gid_map {    /* 64 bytes -- 1 cache line */
        u32 nr_extents;
        union {
                struct uid_gid_extent extent[UID_GID_MAP_MAX_EXTENTS];
                struct {
                        struct uid_gid_extent *forward;
                        struct uid_gid_extent *reverse;
                };
        };
};
```

4.15: Bump Limit of Allowed User Namespace Mappings from 5 to 340



# MAPPINGS	PATCH-V1	PATCH-NEW
0 mappings	158 ns	158 ns
1 mappings	164 ns	157 ns
2 mappings	170 ns	158 ns
3 mappings	175 ns	161 ns
5 mappings	187 ns	165 ns
10 mappings	218 ns	199 ns
50 mappings	528 ns	218 ns
100 mappings	980 ns	229 ns
200 mappings	1880 ns	239 ns
300 mappings	2760 ns	240 ns
340 mappings	not tested	248 ns

4.18: Mounting *Interesting* Filesystem in non-initial User Namespaces

- aa4bf44dc851c6bdd4f7b61b5f2c56c84dfe2ff0
- 6397fac4915ab3002dc15aae751455da1a852f25
- 11a8b9270e16e36d5fb607ba4b60db2958b7c625
- 3edf652fa16562fb57a5a4b996ba72e2d7cdc38b
- d5e7b3c5f51fc6d34e12b6d87bfd30ab277c4625
- ece66133979b211324cc6aff9285889b425243d2
- 3fda0e737e906ce73220b20c27e7f792d0aac6a8
  dbf107b2a7f36fa635b40e0b554514f599c75b33
- c9582eb0ff7d2b560be60eafab29183882cdc82b
- 8cb08329b0809453722bc12aa912be34355bcb66
- 73f03c2b4b527346778c711c2734dbff3442b139
- 57b56ac6fecb05c3192586e4892572dd13d972de
- 593d1ce854dff93b3c9066e897192eb676b09c46
- 55956b59df336f6738da916dbb520b6e37df9fbd ***<- kernel regression***
- 0031181c49ca94b14b11f08e447f40c6ebc842a4
- bc6155d1326092f4c29fe05a32b614249620d88e
- b1d749c5c34112fab5902c43b2a37a0ba1e5f0f1
- f3f1a18330ac1b717cd7a32adff38d965f365aa2
- e45b2546e23c2d10f8585063a15c745a7603fac9
- 4ad769f3c346ec3d458e255548dec26ca5284cf6

4.17: Uevent Injection

- 94e5e3087a67c765be98592b36d8d187566478d5
- 692ec06d7c92af8ca841a6367648b9b3045344fd
- 26045a7b14bc7a5455e411d820110f66557d6589
- a3498436b3a0f8ec289e6847e1de40b4123e1639
- 90d52d4fd82007005125d9a8d2d560a1ca059b9d
- 9d3df886d17b5ef73d4018841ef0a349fcd109ea

5.0: Seccomp Trap To Userspace

- db5113911abaa7eb20cf115d4339959c1aecea95
- a5662e4d81c4d4b08140c625d0f3c50b15786252
- 6a21cc50f0c7f87dae5259f6cfefe024412313f6
- fec7b6690541b8128663a13c9586b1daf42b0a6c

UNSPEAKABLE HORRORS FROM USER SPACE PARALYZE THE LIVING AND RESURRECT THE DEAD!
SECCOMP TRAP TO USER SPACE
with TYCHO ANDERSEN
VAMPIRA
LYLE TALBOT
A J. Edward Reynolds Production
Produced and Directed by Edward D. Wood, Jr.
a DCA release

At some point: New Mount API

- Al Viro’s kernel.org repo
- David Howells’ kernel.org repo

- Range of proposed new syscalls:

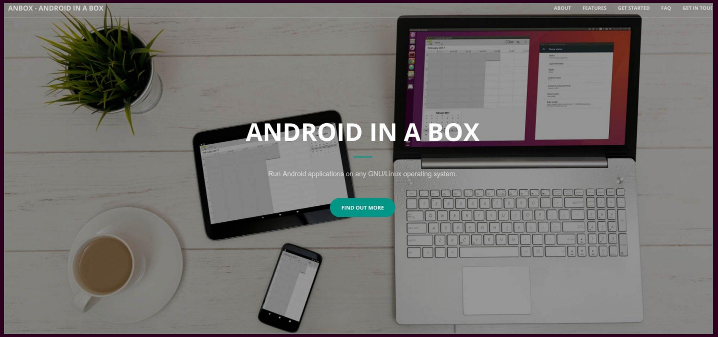
```
-   fspick(int dfd, const char path, unsigned int flags)
-   fsmount(int fs_fd, unsigned int flags, unsigned int attr_flags)
-   fsconfig(int fd, unsigned int cmd, const char _key, const void _value, int aux)
-   fsopen(const char _fs_name, unsigned int flags)
-   move_mount(int from_dfd, const char *from_pathname, int to_dfd,
               const char *to_pathname, unsigned int flags)
-   open_tree(int dfd, const char *filename, unsigned flags)
-   ...
```

At some point: Restricting Path Resolution

- Aleksa’s Github repo
- New flags for open{at}() to restrict path resolution:
  - `O_XDEV`
  - `O_NOMAGICLINKS`
  - `O_NOSYMLINKS`
  - `O_BENEATH`
  - `O_THISROOT`

5.0: Android binderfs Filesystem

- 3ad20fe393b31025bebfc2d76964561f65df48aa
- 3fdd94acd50d607cf6a971455307e711fd8ee16e
- b6c770d7c9dc7185b17d53a9d5ca1278c182d6fa
- 849d540ddfcd4f232f3b2cf40a2e07eccbd6212c
- c13295ad219d8bb0e47942d4cfc8251de449a67e
- 36bdf3cae09df891b191f3955c8e54a2e05d67d0
- 7fefaadd6a962987baac50e7b3c4c3d5ef9b55c6
- 7e7ca7744a539f1a172e3b81c29d000787e3d774
- 6fc23b6ed8fa0ba6cc47b2f8756df1199abc3a5c
- 7d0174065f4903fb0ce0bab3d5047284faa7226d
- 7c4d08fc4d5aca073bd4ebecbb9eda5e4d858b71
- e98e6fa18636609f14a7f866524950a783cf4fbf
- 36975fc3e5f241cc4f45df4ab4624d7d5199d9ed
- 01b3f1fc568352a1ffdcd3ee82a0297f16cc9bd9
- 4198479524aeccaf53c3a4cc73784982535573fa
- 29ef1c8e16aed079ac09989d752e38d412b6e1a8
- 01684db950ea2b840531ab9298a8785776b6f6e8



At some point: File Descriptors for Processes

- My kernel.org repo
- First proposed syscall:
  - `pidfd_send_signal(int, pidfd, int sig, siginfo_t info, unsigned int flags)`

A Year of Container Kernel work

Past, Present, and Future of Container Kernel Features

FOSDEM, 2019
Brussels, Belgium

Christian Brauner
christian@brauner.io
christian.brauner@ubuntu.com
@brau_ner
https://brauner.io/