14th ANNUAL WORKSHOP 2018

JOURNEY TO VERBS IOCTL

Michael J. Ruhl, Software Developer

Intel Corporation

March 29, 2018

LEGAL NOTICES AND DISCLAIMERS

- **Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.**
- **No computer system can be absolutely secure.**
- **Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.**
- **Intel, the Intel logo, Xeon and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.**
- **© 2018 Intel Corporation.**

TOPICS

- **Background**
- **Code Base**
- **Compiler gotchas**
- **Data passing nuance or the PTR_IN Optimization**
- **Linker Issues**
- **My First Real Problem**
- **Naming convention**
- **The Legacy Interface**
- **Summing up**

BACKGROUND

SECURITY CONCERN WITH CURRENT ABI

CVE-2016-4565

A flaw was found in the way certain interfaces of the Linux kernel's Infiniband subsystem used write() as bi-directional ioctl() replacement, which could lead to insufficient memory security checks when being invoked using the splice() system call. A local unprivileged user on a system with either Infiniband hardware present or RDMA Userspace Connection Manager Access module explicitly loaded, could use this flaw to escalate their privileges on the system.

Source: https://access.redhat.com/security/cve/CVE-2016-4565

From: Jason Gunthorpe <jgunthorpe at obsidianresearch.com>

The drivers/infiniband stack uses write() as a replacement for bi-directional ioctl(). This is not safe. There are ways to trigger write calls that result in the return structure that is normally written to user space being shunted off to user specified kernel memory instead.

For the immediate repair, detect and deny suspicious accesses to the write API.

For long term, update the user space libraries and the kernel API to something that doesn't present the same security vulnerabilities (likely a structured ioctl() interface).

The impacted uAPI interfaces are generally only available if hardware from drivers/infiniband is installed in the system.

THE WORK AROUND

```
commit e6bd18f57aad1a2d1ef40e646d03ed0f2515c9e3
Author: Jason Gunthorpe <jgunthorpe@obsidianresearch.com>
Date:   Sun Apr 10 19:13:13 2016 -0600

IB/security: Restrict use of the write() interface
For the immediate repair, detect and deny suspicious accesses to the write API.


static inline bool ib_safe_file_access(struct file *filp)
{
       return filp->f_cred == current_cred() && segment_eq(get_fs(), USER_DS);
}
```

THE SOLUTION

- **Keep write semantics, use IOCTLs for control**
- **Presented as:**
  - Kernel ABI_Mbarak OpenFabrics Alliance Workshop 2017
- **The Verbs IOCTL interface was introduced with this patch:**
  - IB/core: Add new ioctl interface

FIRST THINGS FIRST

CHOOSE THE RIGHT CODE BASE

- **The latest code is in the tree, but when I started….**
  - [PATCH RFC 00/10] IB/core: SG IOCTL based RDMA ABI
- **Vs (merged into the tree):**
  - [PATCH V1 00/13] IB/core: SG IOCTL based RDMA ABI
- **Using the right code base solves lots of problems...**
- **There have been many recent updates as well. Some of these are pending, others have been merge.**
- **Monitoring the mailing list is very important!**

MONITOR THE MAILING LIST

For instance

Author: Matan Barak <matanb@mellanox.com>

Date: Tue Jan 23 15:55:04 2018 -0500

IB/uverbs: Fix method merging in uverbs_ioctl_merge

Fix a bug in uverbs_ioctl_merge that looked at objects iterators number instead of methods iterators number when merging methods.

While we're at it, make the uverbs_ioctl_merge code a bit more clear and faster.

- **My code crashed without this patch. This patch (or something similar) is available in 4.16.**
- **So remember that this is code in progress. It is getting better every day, so don't forget to watch the mailing list.**

A bug in my code (mis-used enumeration) revealed a crash in the original code.

COMPILER GOTCHAS

COMPILER FUN

- **The new interface is based on a MACRO defined "domain specific language". MACRO errors lead to interesting compiler issues.**
- **Here are some common compile issues I ran into with the MACROs used to describe the objects/methods/attributes.**
  - Minimum size data structure required
  - Inconsistent sizeof() usage
  - Missing '&'
  - Missing parameter

MINIMUM SIZE DATA STRUCTURE REQUIRED

```
CC [M]  drivers/infiniband/hw/hfi1/uverbs_objects.o
In file included from ./include/rdma/rdma_vt.h:61:0,
                 from drivers/infiniband/hw/hfi1/hfi.h:72,
                 from drivers/infiniband/hw/hfi1/uverbs_objects.c:47:
./include/rdma/uverbs_types.h:161:36: error: expected expression before ‘<’ token
    UVERBS_BUILD_BUG_ON((_obj_size) < sizeof(struct ib_uobject_file)), \
                                    ^
./include/rdma/uverbs_ioctl.h:256:17: note: in definition of macro ‘_UVERBS_OBJECT’
   .type_attrs = _type_attrs,     \
                 ^
drivers/infiniband/hw/hfi1/uverbs_objects.c:172:1: note: in expansion of macro
‘DECLARE_UVERBS_OBJECT’
 DECLARE_UVERBS_OBJECT(hfi1_object_psm_fd,
 ^
./include/rdma/uverbs_types.h:161:4: note: in expansion of macro ‘UVERBS_BUILD_BUG_ON’
    UVERBS_BUILD_BUG_ON((_obj_size) < sizeof(struct ib_uobject_file)), \
    ^
drivers/infiniband/hw/hfi1/uverbs_objects.c:174:10: note: in expansion of macro
‘UVERBS_TYPE_ALLOC_FD’
         &UVERBS_TYPE_ALLOC_FD(0, u64,
```

UVERBS_TYPE_ALLOC_FD requires a data structure with a minimum size of struct ib_uobject_file

```
DECLARE_UVERBS_OBJECT(hfi1_object_psm_fd,
                      HFI1_OBJECT_PSM_FD,
-                     &UVERBS_TYPE_ALLOC_FD(0, u64,
+                     &UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
                                           hfi1_psm_fd_handler,
                                           &hfi1_psm_fd_fops,
                                           "[hfi1_psm_fd]", O_RDWR),
```

In this case, the data structure will be a struct ib_uobject_file plus your data. So this is the minimum size.

The data structures are layered. Make sure you understand where your data structure resides in the layers.

_PTR_IN/_PTR_OUT: Do NOT use sizeof()

- **Some MACROs need a sizeof():**

  &UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file), hfi1_psm_file,
  &hfi1_file_ops, "[psm_file_ops]", O_RDWR),

- **Some do not (it is embedded in the MACRO):**

  &UVERBS_ATTR_PTR_IN(HFI1_DEV_HDR, struct ib_uverbs_cmd_hdr,
  UA_FLAGS(UVERBS_ATTR_SPEC_F_MANDATORY)),

- **If a sizeof() is used on the struct ib_uverbs_cmd_hdr, the incorrect size will be determined for the memcpy() and/or allocation.**

Not really a compiler issue, but a language issue. There are new MACRO wrappers for the types. UVERBS_ATTR_TYPE(<data type>). I am not sure if they address this issue.

WHERE DID MY '&' GO…

```
CC [M]  drivers/infiniband/hw/hfi1/uverbs_device.o
In file included from drivers/infiniband/hw/hfi1/uverbs_obj.h:51:0,
                 from drivers/infiniband/hw/hfi1/uverbs_device.c:48:
./include/rdma/uverbs_types.h:161:52: error: incompatible types when initializing type 'const struct uverbs_obj_type *' using type 'const struct uverbs_obj_type'
    UVERBS_BUILD_BUG_ON((_obj_size) < sizeof(struct ib_uobject_file)), \
                                      ^
./include/rdma/uverbs_ioctl.h:289:17: note: in definition of macro '_UVERBS_OBJECT'
   .type_attrs = _type_attrs,     \
                 ^
drivers/infiniband/hw/hfi1/uverbs_device.c:280:1: note: in expansion of macro 'DECLARE_UVERBS_OBJECT'
 DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
 ^
./include/rdma/uverbs_types.h:161:4: note: in expansion of macro 'UVERBS_BUILD_BUG_ON'
    UVERBS_BUILD_BUG_ON((_obj_size) < sizeof(struct ib_uobject_file)), \
    ^
drivers/infiniband/hw/hfi1/uverbs_device.c:281:9: note: in expansion of macro 'UVERBS_TYPE_ALLOC_FD'
        UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
        ^
```

‘&’ is required for a lot of the definitions. Make sure you don’t miss any.

```
DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
-                     UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
+                     &UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
                                            hfi1_psm_file,
                                            &hfi1_file_ops,
                                            "[psm_file_ops]", O_RDWR),
```

WHERE'S MY '&' CONTINUED…

```
CC [M]  drivers/infiniband/hw/hfi1/uverbs_device.o
In file included from drivers/infiniband/hw/hfi1/uverbs_obj.h:51:0,
                 from drivers/infiniband/hw/hfi1/uverbs_device.c:48:
./include/rdma/uverbs_ioctl.h:284:24: error: incompatible types when initializing type 'const struct uverbs_method_def *'
using type 'const struct uverbs_method_def'
  (sizeof((const struct uverbs_method_def * const []){__VA_ARGS__}) / \
./include/rdma/uverbs_ioctl.h:290:18: note: in expansion of macro '_UVERBS_OBJECT_METHODS_SZ'
   .num_methods = _UVERBS_OBJECT_METHODS_SZ(__VA_ARGS__),  \
./include/rdma/uverbs_ioctl.h:294:3: note: in expansion of macro '_UVERBS_OBJECT'
   _UVERBS_OBJECT(_id, _type_attrs, ##__VA_ARGS__)
drivers/infiniband/hw/hfi1/uverbs_device.c:280:1: note: in expansion of macro 'DECLARE_UVERBS_OBJECT'
 DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
./include/rdma/uverbs_ioctl.h:291:29: error: incompatible types when initializing type 'const struct uverbs_method_def *'
using type 'const struct uverbs_method_def'
   .methods = &(const struct uverbs_method_def * const []){__VA_ARGS__} })
./include/rdma/uverbs_ioctl.h:294:3: note: in expansion of macro '_UVERBS_OBJECT'
   _UVERBS_OBJECT(_id, _type_attrs, ##__VA_ARGS__)
drivers/infiniband/hw/hfi1/uverbs_device.c:280:1: note: in expansion of macro 'DECLARE_UVERBS_OBJECT'
 DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
```

The subtlety of the missing '&'

```
@@ -282,7 +282,7 @@ static DECLARE_UVERBS_METHOD(
                                hfi1_psm_file,
                                &hfi1_file_ops,
                                "[psm_file_ops]", O_RDWR),
-                       hfi1_fd_create);
+                       &hfi1_fd_create);
```

MISSING PARAMETERS

```
CC [M]  drivers/infiniband/hw/hfi1/uverbs_device.o
In file included from drivers/infiniband/hw/hfi1/uverbs_obj.h:51:0,
                 from drivers/infiniband/hw/hfi1/uverbs_device.c:48:
./include/rdma/uverbs_ioctl.h:189:27: error: expected expression before '.' token
#define UA_FLAGS(_flags)  .flags = _flags
./include/rdma/uverbs_ioctl.h:266:52: note: in definition of macro
'_UVERBS_METHOD_ATTRS_SZ'
  (sizeof((const struct uverbs_attr_def * const []){__VA_ARGS__}) /\
./include/rdma/uverbs_ioctl.h:277:3: note: in expansion of macro '_UVERBS_METHOD'
_UVERBS_METHOD(_id, _handler, 0, ##__VA_ARGS__)
....
./include/rdma/uverbs_ioctl.h:277:38: note: in expansion of macro 'UVERBS_ATTR_FD'
   _UVERBS_METHOD(_id, _handler, 0, ##__VA_ARGS__)
./include/rdma/uverbs_ioctl.h:277:38: note: in expansion of macro 'UA_FLAGS'
drivers/infiniband/hw/hfi1/uverbs_device.c:267:8: note: in expansion of macro
'DECLARE_UVERBS_METHOD'
 static DECLARE_UVERBS_METHOD(
```

Which parameter is missing?

```
static DECLARE_UVERBS_METHOD(
        hfi1_fd_create, HFI1_PSM_FD, hfi1_fd_create_handler,
-       &UVERBS_ATTR_FD(HFI1_PSM_CREATE_FD,
+       &UVERBS_ATTR_FD(HFI1_PSM_CREATE_FD, HFI1_OBJECT_FD,
                        UVERBS_ACCESS_NEW,
                        UA_FLAGS(UVERBS_ATTR_SPEC_F_MANDATORY)));
```

ANOTHER MISSING PARAMETER

```
CC [M]  drivers/infiniband/hw/hfi1/uverbs_device.o
drivers/infiniband/hw/hfi1/uverbs_device.c:284:24: error: macro "UVERBS_TYPE_ALLOC_FD" requires 6 arguments, but only 5 given
        &hfi1_fd_create);
In file included from drivers/infiniband/hw/hfi1/uverbs_obj.h:51:0,
                 from drivers/infiniband/hw/hfi1/uverbs_device.c:48:
drivers/infiniband/hw/hfi1/uverbs_device.c:281:10: error: ‘UVERBS_TYPE_ALLOC_FD’ undeclared here (not in a function)
        &UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
./include/rdma/uverbs_ioctl.h:289:17: note: in definition of macro ‘_UVERBS_OBJECT’
   .type_attrs = _type_attrs,     \
drivers/infiniband/hw/hfi1/uverbs_device.c:280:1: note: in expansion of macro ‘DECLARE_UVERBS_OBJECT’
 DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
drivers/infiniband/hw/hfi1/uverbs_device.c:175:12: warning: ‘hfi1_psm_file’ defined but not used [-Wunused-function]
 static int hfi1_psm_file(struct ib_uobject_file *uobj_file,
```

UVERBS_TYPE_ALLOC_FD missing parameter

```
DECLARE_UVERBS_OBJECT(hfi1_object_fd, HFI1_OBJECT_FD,
                      &UVERBS_TYPE_ALLOC_FD(0, sizeof(struct ib_uobject_file),
-                                           hfi1_psm_file,
+                                           hfi1_psm_file,
                                            &hfi1_file_ops,
                                            "[psm_file_ops]", O_RDWR),
                      &hfi1_fd_create);
```

DATA STRUCTURE LAYERING

Most data structures will need to have the ib_object embedded in them. Access is via container_of().

- I.e:

```
struct my_private_object {
    struct ib_uobject uobject;
    struct my_ptr     *ptr;
    struct list_head  my_list;
};

static int my_private_free(struct ib_uobject *uobj,
               enum rdma_remove_reason why)

struct my_private_object *ptr =
        container_of(uobj, struct ib_ucq_object, uobject);
```

#define container_of(ptr, type, member)

DATA PASSING NUANCE OR THE PTR_IN OPTIMIZATION

Data or data structures that are sizeof(attr->data) MUST be copied directly to the attributes data member.

```
static void set_attr(struct ib_uverbs_attr *attr, __u16 attr_id, __u16
                     len, __u16 flags, void *data, enum attr_dir dir)
{
     attr->attr_id = attr_id;
     attr->len = len;
     attr->flags = flags;

     switch (dir) {
     case PTR_IN:
          if (len <= sizeof(attr->data) && data)
               memcpy(&attr->data, data, len);
          else
               attr->data = (__u64) data;
          break;
     …
     }
}
```

This is a helper function I wrote in my user space test app. The UVERBS kernel inline uverbs_copy_from() checks the length, and then either does either a memcpy() of the data element, or a copy_from_user() using data as a pointer.

LINKER ISSUES

HERE COMES THE LINKER!

- **When I went to link, the following symbols where not available and needed to be exported:**
  - uverbs_default_objects
  - uverbs_idr_class
  - uverbs_fd_class
  - uverbs_close_fd
  - uverbs_uobject_get
  - uverbs_uobject_set
- **Most likely there will be more symbols from the rdma_core.c module that will need to be exported over time**
- **Current patches on the mailing list have exported some of these symbols**

MY FIRST REAL PROBLEM

THE UNEXECTED EINVAL…

My first attempt to call the ioctl

- **The IOCTL interface is “done”, but the feature set is incomplete**
- **No examples for test code to use the interface**
- **Example code for new methods consists of:**
  - uverbs_create_cq_handler()
  - uverbs_destroy_cq_handler()
- **New methods have been added, or will be added soon**

I wrote my on user space test code from examining the above examples.

My first test was…

```
int test_ioctl()
{
 …
    hdr = req;
    attr = req + sizeof(*hdr);
    hdr->length = len;
    hdr->num_attrs = 1;
    hdr->object_id = UVERBS_UDATA_DRIVER_DATA_FLAG + 1;
    hdr->method_id = 0;
    set_attr(attr, 0, 0, 0, 0, NONE);
    ret = ioctl(fd, RDMA_VERBS_IOCTL, req);
 …
}
```

ret = -EINVAL!

User space test code. fd = open(“/dev/infiniband/uverbs”, O_RDWR)’

After some sleuthing…

- **core/uverbs_ioctl.c**

```
if ((method_spec->flags & UVERBS_ACTION_FLAG_CREATE_ROOT) ^ !file->ucontext)
     return -EINVAL;
```

- **file->ucontext == NULL**
- **The UVERBS_ACTION_FLAG_CREATE_ROOT is only used in:**
  - DECLARE_UVERBS_CTX_METHOD()
- **This is not used in any of the example code**
- **Question: So where was the ucontext supposed to be created?**
- **Answer: Via the old write() interface**

MY SOLUTION

Re-use the orginal ib_uverbs_get_context()

- **Factor out the cmd interface information that was necessary for the write() command.**
- **Implemented necessary ioctl() attributes (response only)**
- **Determined common code**
- **Added new code path**

Create a get context handler using the IOCTL interface

```
static DECLARE_UVERBS_CTX_METHOD(
        uverbs_object_device_ctx, UVERBS_DEV_CTX,
        uverbs_get_ctx_handler, 0,
        &UVERBS_ATTR_PTR_OUT(UVERBS_DEV_CTX_RESP,
                             struct ib_uverbs_get_context_resp,
                             UA_FLAGS(UVERBS_ATTR_SPEC_F_MANDATORY)),
       &uverbs_uhw_compat_in, &uverbs_uhw_compat_out);

DECLARE_UVERBS_OBJECT(uverbs_object_device, UVERBS_OBJECT_DEVICE,
                      &UVERBS_TYPE_ALLOC_IDR(0, uverbs_free_ctx),
                      &uverbs_object_device_ctx);
```

I selected the standard DEVICE object for the method. This object currently has no methods (not sure if it should). However, creating a new object for creating contexts didn't seem correct (since I am creating the context on for the device), but this may not be the correct way to do this. I am sure there will be some discussion on this…

IOCTL interface handler for creating contexts

```
static int uverbs_get_ctx_handler(struct ib_device *ib_dev,
                                  struct ib_uverbs_file *file, struct uverbs_attr_bundle *attrs)
{
        struct ib_udata uhw;
        const struct uverbs_attr *attr;
        int ret;

        if (file->ucontext)
                return -EINVAL;

        if (!(ib_dev->uverbs_cmd_mask & 1ULL << IB_USER_VERBS_CMD_GET_CONTEXT))
                return -EOPNOTSUPP;

        attr = uverbs_attr_get(attrs, UVERBS_DEV_CTX_RESP);
        if (IS_ERR(attr))
                return PTR_ERR(attr);

        create_udata(attrs, &uhw);

        ret = uverbs_get_context(file, ib_dev, &uhw, attr->ptr_attr.data);

        return ret;
}
```

create_udata() may not be needed after some of the latest patches. Still exploring this.

NAMING CONVENTIONS

YOUR NAMING CONVENTION IS CRITICAL!

- **Make sure all of your enumerations (object, method, attribute) are named correctly**
- **If you are cutting and pasting the definitions, double, triple, and quadruple check your enumeration types**
- **It might be possible that the language macros can do type checking at some point. Although, I am not sure how this would work**
- **Recent patches to the core have refined its naming convention**

Enumeration Naming Example…

```
enum hfi1_objects {
  HFI1_OBJECT_DEVICE = UVERBS_UDATA_DRIVER_DATA_FLAG,
  HFI1_OBJECT_PSM_LEGACY,
  HFI1_OBJECT_LAST,
};
enum hfi1_psm_method_ids {
  HFI1_METHOD_PSM_FD_CREATE,
};
enum hfi1_psm_fd_attr_ids {
  HFI1_ATTR_PSM_FD_CREATE,
};
```

LEGACY HFI1/PSM SUPPORT

LEGACY HFI1/PSM INTERFACE

Supporting the OId interface with the new IOCTL interface

- **Define the OBJECTs**
- **Define the METHODs**
- **Define the ATTRIBUTEs**
- **Code away….**

THE OBJECTS

```
DECLARE_UVERBS_OBJECT(hfi1_object_device, HFI1_OBJECT_DEVICE,
                      &UVERBS_TYPE_ALLOC_IDR(0, hfi1_object_dev_cap),
                       &hfi1_dev_capabilities,  &hfi1_psm_capabilities);

DECLARE_UVERBS_OBJECT(hfi1_object_psm_legacy, HFI1_OBJECT_PSM_LEGACY,
                      &UVERBS_TYPE_ALLOC_FD(0,   sizeof(struct hfi1_psm_legacy_file),
                                            hfi1_psm_fd_handler, &hfi1_psm_fd_fops,
                                            "[hfi1_psm_fd]", O_RDWR),
                      &hfi1_psm_fd_create);

DECLARE_UVERBS_OBJECT_TREE(hfi1_uverbs_objects,
                           &hfi1_object_device,
                           &hfi1_object_psm_legacy);
```

METHODS

hf1_psm_fd_create_handler is the new ".open"

```
static DECLARE_UVERBS_METHOD(hfi1_psm_fd_create,
        HFI1_METHOD_PSM_FD_CREATE,
        hfi1_psm_fd_create_handler,
        &UVERBS_ATTR_FD(HFI1_ATTR_PSM_FD_CREATE,
                HFI1_OBJECT_PSM_LEGACY,
                UVERBS_ACCESS_NEW,
                UA_FLAGS(UVERBS_ATTR_SPEC_F_MANDATORY)));
```

FILE_OPS

Updated the ".release", everything from the original HFI1 file_ops

```
static const struct file_operations hfi1_psm_fd_fops = {
        .owner = THIS_MODULE,
        .write_iter = hfi1_uverbs_write_iter,
-       .open = hfi1_file_open,
-       .release = hfi1_file_close,
+       .release = hfi1_psm_fd_close,
        .unlocked_ioctl = hfi1_uverbs_file_ioctl,
        .poll = hfi1_uverbs_poll,
        .mmap = hfi1_uverbs_file_mmap,
        .llseek = noop_llseek,
};
```

THE CODE

The new ".open"

```
static int hfi1_psm_fd_create_handler(struct ib_device *ib_dev,
                                      struct ib_uverbs_file *file,
                                      struct uverbs_attr_bundle *attrs)
{
      const struct uverbs_attr *attr;
      struct rvt_dev_info *rdi = ib_to_rvt(ib_dev);
      struct hfi1_ibdev *verbs_dev = dev_from_rdi(rdi);
      struct hfi1_devdata *dd = dd_from_dev(verbs_dev);
      struct ib_uobject *uobj;
      struct ib_uobject_file *uobj_file;
      struct file *fp;

*     attr = uverbs_attr_get(attrs, HFI1_ATTR_PSM_FD_CREATE);
*     if (IS_ERR(attr))
*           return PTR_ERR(attr);

*     uobj = attr->obj_attr.uobject;
      uobj_file = container_of(uobj, struct ib_uobject_file, uobj);
      fp = uobj_file->uobj.object;

      return hfi1_file_open(dd, fp, fp->private_data);
}
```

Verbs FDs use the private_data. The HFI1 driver creates its own private data. This was the only major difficulty. So I did have to get a little creative with that.

A recent patch has put the '*' lines in an inline function.

THE CODE

The new ".release"

```
static int hfi1_psm_fd_close(struct inode *inode, struct file *fp)
{
        extern void uverbs_close_fd(struct file *f);
        struct hfi1_filedata *fd = fp->private_data;
        void *pd = fd->extra;
        int ret = 0;

        ret = hfi1_file_close(inode, fp);

        /* restore verbs private data */
        fp->private_data = pd;
        uverbs_close_fd(fp);

        return ret;
}
```

Clean up private data so uverbs gets its private data back.

SUMMING UP

- **Monitor the mailing list for new updates**
  - The feature set is not yet complete
  - This code is evolving
  - Bug fixes and new features are still coming, Matan Barak is driving most of this
- **The MACRO language can be challenging, but is pretty straight forward**
- **Make sure your enumerations are well named, and are used in the right places (be very careful with cut and paste)**
- **Understanding how the data structures are layered is very important.**
- **More access to the RDMA core (EXPORT_SYMBOL) will be necessary**

The provider interface (Jason Gunthorpe) is in progress. Supports the IOCTL and write() conventions.

QUESTIONS/COMMENTS

NOTICES AND DISCLAIMERS

OPTIMIZATION NOTICE

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

14th ANNUAL WORKSHOP 2018

THANK YOU

Michael J. Ruhl, Software Developer

Intel Corporation