Linux XIA source code - XID mapping

static inline int xt_to_vxt_rcu(xid_type_t ty): Convert XID type to its Virtual XID Type

Called in files: fib.c, route.c, socket.c

flow.c:struct xip_ppal_ctx *xip_find_ppal_ctx_rcu(struct net *net, xid_type_t ty);

route.c: static inline struct xip_route_proc *get_an_rproc_rcu(const xid_type_t ty);

socket.c: static int xia_create(struct net *net, struct socket *sock, int protocol, int kern);

static inline int xt_to_vxt(xid_type_t ty): same with RCU

Called in files: fib.c, route.c, socket.c

flow.c: int xip_add_ppal_ctx(struct net *net, struct xip_ppal_ctx *ctx);
int xip_del_ppal_ctx(struct net *net, struct xip_ppal_ctx *ctx);

route.c: int xip_add_router(struct xip_route_proc *rproc);
int xip_del_router(struct xip_route_proc *rproc);

socket.c: int xia_add_socket(struct xia_socket_proc *sproc);
int xia_del_socket_begin(struct xia_socket_proc *sproc)

int vxt_register_xidty(xid_type_t ty);
int vxt_unregister_xidty(xid_type_t ty);

When you compile your kernel, __KERNEL__ is defined on the command line. User-space programs need access to the kernel headers, but some of the info in kernel headers is intended only for the kernel. Wrapping some statements in an #ifdef __KERNEL__/#endif block ensures that user-space programs don't see those statements.

xia_vxidty.h

1. likely()
2. rcu_dereference()
3. __be32_to_cpu()

vxidty.c

1. static **DEFINE_MUTEX**(vxt_mutex);
2. **__read_mostly**
3. **BUILD_BUG_ON_NOT_POWER_OF_2**(XIP_VXT_TABLE_SIZE);

4. **BUILD_BUG_ON**(XIP_VXT_TABLE_SIZE < XIP_MAX_XID_TYPES);
5. ret = **find_first_zero_bit**(allocated_vxt, XIP_MAX_XID_TYPES);
6. **__set_bit**(ret, allocated_vxt); /* Cook a new map. */
7. **memmove**(new_map, old_map, MAP_SIZE_IN_BYTE);
8. **BUG_ON**(!__test_and_clear_bit(entry->index, allocated_vxt));

```
#include <assert.h>
#define BUG_ON(b) assert(!(b))
#define BUILD_BUG_ON(condition) ((void)sizeof(char[1 - 2*!!(condition)]))
#define __force

#define XIA_XID_MAX 20; //xia.h
int xip_init_ppal_ctx(struct xip_ppal_ctx *ctx, xid_type_t ty) {
	ctx->xpc_ppal_type = ty;
	ctx->xpc_xtbl = NULL;
	xdst_init_anchor(&ctx->negdep);
	return 0;
}

int xip_add_ppal_ctx(struct net *net, struct xip_ppal_ctx *ctx)
{
	xid_type_t ty = ctx->xpc_ppal_type;
	int vxt = xt_to_vxt(ty);

	if (unlikely(vxt < 0))
		return -EINVAL;

	if (net->xia.fib_ctx[vxt]) {
		BUG_ON(net->xia.fib_ctx[vxt]->xpc_ppal_type != ty);
		return -EEXIST;
	}
	rcu_assign_pointer(net->xia.fib_ctx[vxt], ctx);

	return 0;
}

struct xip_ppal_ctx *xip_del_ppal_ctx(struct net *net, xid_type_t ty)
{
	int vxt = xt_to_vxt(ty);
	struct xip_ppal_ctx *ctx;

	BUG_ON(vxt < 0);
	ctx = net->xia.fib_ctx[vxt];
	BUG_ON(!ctx);
```

```
	BUG_ON(ctx->xpc_ppal_type != ty);
	RCU_INIT_POINTER(net->xia.fib_ctx[vxt], NULL);
	synchronize_rcu();
	return ctx;
}

struct xip_ppal_ctx *xip_find_ppal_ctx_rcu(struct net *net, xid_type_t ty)
{
	int vxt = xt_to_vxt_rcu(ty);

	return likely(vxt >= 0)
		? xip_find_ppal_ctx_vxt_rcu(net, vxt)
		: NULL;
}

/** xip_find_ppal_ctx_vxt_rcu - Find context of principal of virtual type @vxt.
 *
 * RETURN
 *	It returns the struct on success, otherwise NULL.
 *
 * NOTE
 *	Caller must hold an RCU read lock.
 *
 *	If the caller must keep the reference after an RCU read lock,
 *	it must call xtbl_hold before releasing the RCU lock.
 */
static inline struct xip_ppal_ctx *xip_find_ppal_ctx_vxt_rcu(struct net *net,
	int vxt)
{
	return rcu_dereference(net->xia.fib_ctx[vxt]);
}
```

IMPORTANT DEFINITION

net_namespace.h: struct net {

```
……
#if defined(CONFIG_XIA) || defined(CONFIG_XIA_MODULE)
	struct netns_xia	xia;
#endif
……
}
```

netns/xia.h:

```
#ifndef __NETNS_XIA_H__
#define __NETNS_XIA_H__

/*
 * XIA's net namespace
 */

#include <net/dst_ops.h>

/* Maximum number of XID types recognized at the same time. */
#define XIP_MAX_XID_TYPES       8

/* It must be a power of 2. */
#define XIP_DST_TABLE_SIZE      256

/* XXX This data structure is definitely not perfect because
 * it does not reflect the load/capacity of a given namespace (struct net),
 * however, it's not clear how it should be shaped since XIA is too
 * young to have any usage data.
 */
struct xip_dst_table {
        struct dst_entry        *buckets[XIP_DST_TABLE_SIZE];
        atomic_t                last_bucket; /* Used for garbage collection. */
};

struct netns_xia {
        /* Hash of principal contexts.
         * Principals that need to link data to struct net, should do so using
         * struct xip_ppal_ctx. It avoids messing the struct netns_xia, and
         * simplifies loading and unloading of principals.
         */
        struct xip_ppal_ctx  *fib_ctx[XIP_MAX_XID_TYPES];

        /* Route cache. */
        struct dst_ops          xip_dst_ops;
        struct xip_dst_table    xip_dst_table;
};
#endif /* __NETNS_XIA_H__ */
```

Builtin GCC Functions - __builtin_clz(); __builtin_ctz(); __builtin_popcount();

1. **int __builtin_clz (unsigned int x)**
This builtin method is to **count the number of leading zero's** in variable. For example, 00000000 00000000 00000000 00010000, Counting the number of leading zero's is 27.

2. **int __builtin_ctz (unsigned int x)**
This builtin method by GCC determines the count of trailing zero in the binary representation of a number. For example, 00000000 00000000 00000000 00010000, Counting the number of trailing zero's is 4.

3. **int __builtin_popcount (unsigned int x)**
This builtin method by GCC determines the number of one's in the binary representation of a number. For example, 00000000 00000000 00000000 00010000, Counting the number of one's is just 1.

4. Example:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num = 16;
   int clz = 0;
   int ctz = 0;
   int ones = 0;

   clz = __builtin_clz(num);
   printf("Number of leading zero's in %d is %d\n", num, clz);

   clz = __builtin_clz(-num);
   printf("Number of leading zero's in %d is %d\n", -num, clz);

   ctz = __builtin_ctz(num);
   printf("Number of trailing zero's in %d is %d\n", num, ctz);

   ones = __builtin_popcount(num);
   printf("Number of one's in %d is %d\n", num, ones);

   return 0;
}
```

bool __sync_bool_compare_and_swap (type *ptr, type oldval type newval, ...)
type __sync_val_compare_and_swap (type *ptr, type oldval type newval, ...)
These builtins perform an atomic compare and swap. That is, if the current value of *ptr is oldval, then write newval into *ptr.
The “bool” version returns true if the comparison is successful and newval was written. The “val” version returns the contents of *ptr before the operation.
__sync_synchronize (...)
This builtin issues a full memory barrier.
type __sync_lock_test_and_set (type *ptr, type value, ...)
This builtin, as described by Intel, is not a traditional test-and-set operation, but rather an atomic exchange operation. It writes value into *ptr, and returns the previous contents of *ptr.
Many targets have only minimal support for such locks, and do not support a full exchange operation. In this case, a target may support reduced functionality here by which the *only* valid value to store is the immediate constant 1. The exact value actually stored in *ptr is implementation defined.
This builtin is not a full barrier, but rather an acquire barrier. This means that references after the builtin cannot move to (or be speculated to) before the builtin, but previous memory stores may not be globally visible yet, and previous memory loads may not yet be satisfied.

XDP U4ID example

1. Command

```
# ruby xlxc-net.rb -n xia -s 2 --create -t star
# modprobe xia_ppal_u4id
# modprobe xia_ppal_xdp

# tcpdump -i xia0br -w u4id_capture.log
 (follow sections below to send packets; once sent, quit tcpdump using Ctrl + C)
 # wireshark u4id_capture.log

# ruby xlxc-start.rb -n xia0
root@xia0:~# xip u4id add 10.0.1.2 0x35d5 -tunnel
root@xia0:~# eserv datagram xip server_address.txt
root@xia0:~# cat server_address.txt
u4id-0a00010235d5000000000000000000000000000-1:
xdp-007f4e38904e83634acc7e1340ef7665e3f1f57a-0

# ruby xlxc-start.rb -n xia1
root@xia1:~# xip u4id add 10.0.4.2 0x35d5 -tunnel
root@xia1:~# ecli datagram xip client_address.txt server_address.txt
root@xia1:~# cat client_address.txt
u4id-0a00040235d5000000000000000000000000000-1:
xdp-007f4e38904e83634acc7e1340ef7665e3f1f57b-0
```

2. Experimental results

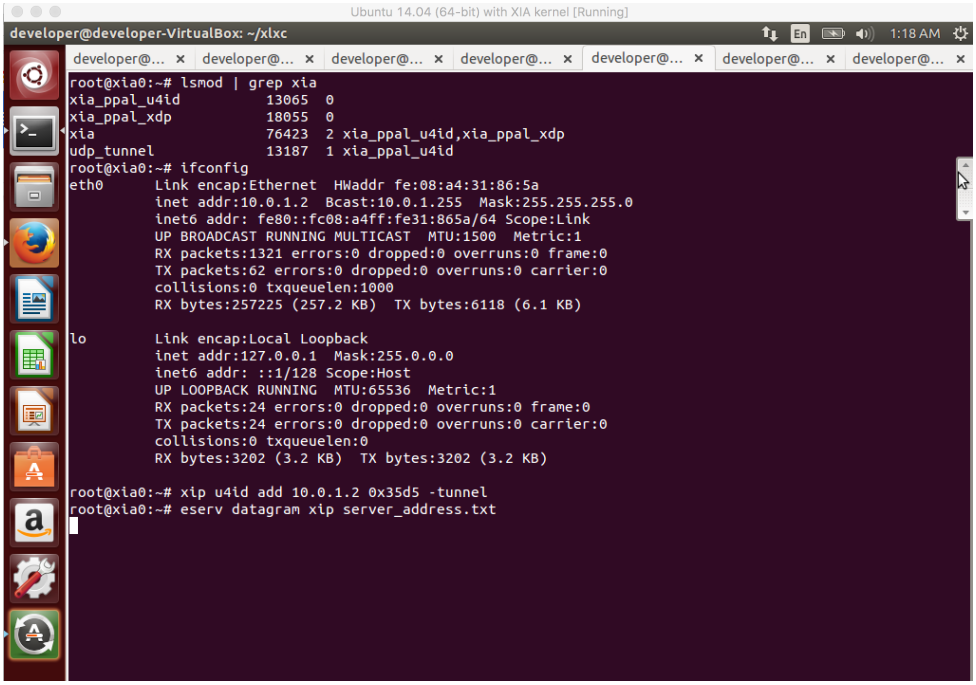

Fig 1. Server

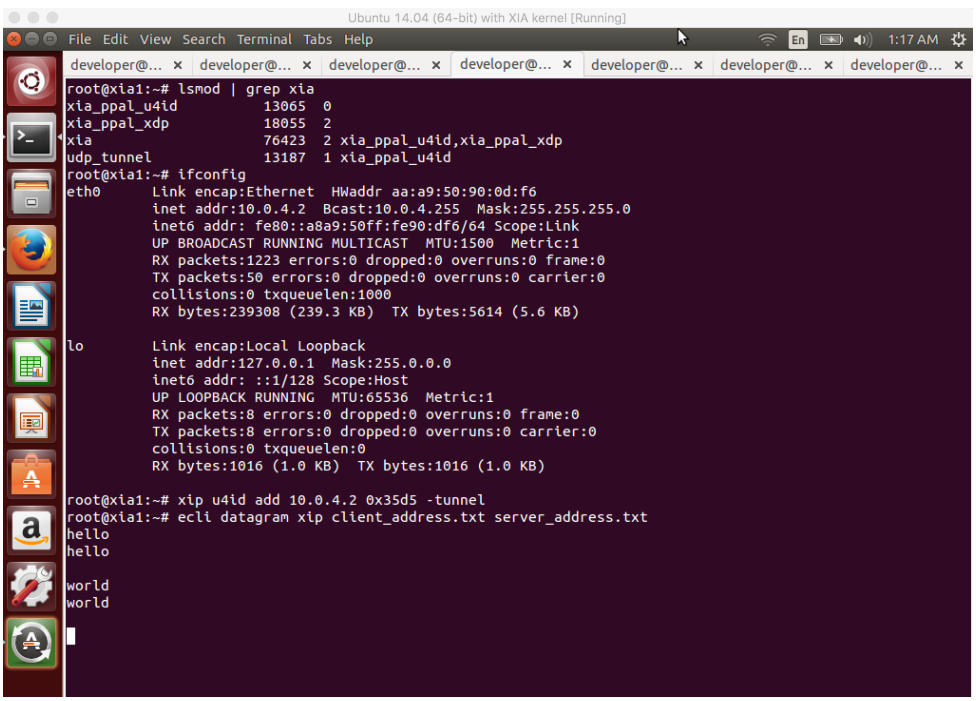

Fig 2. Client

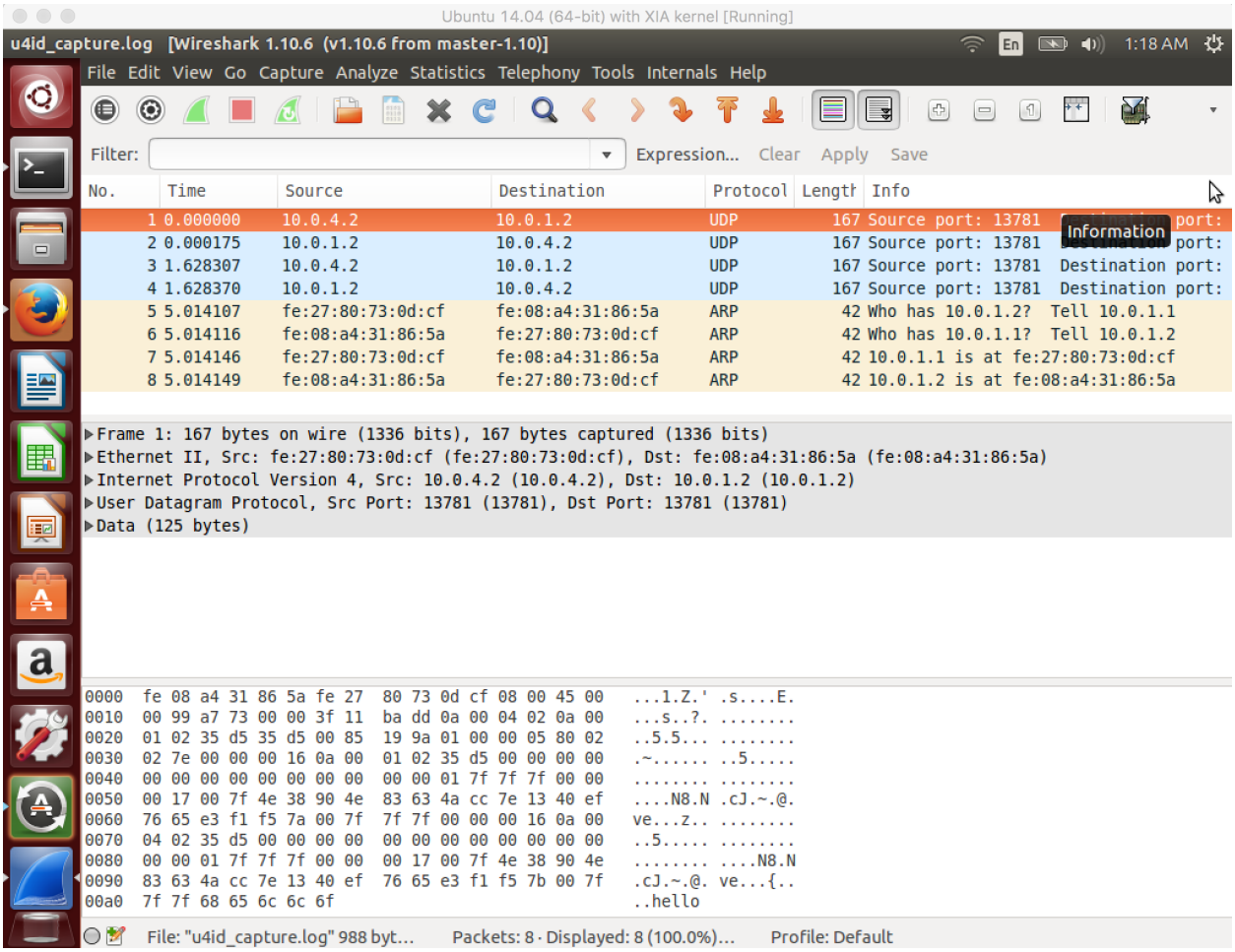

Fig 3. Packets capture

3. Dmesg sample output

[6914.969532] NET: Registered protocol family 41
[6914.969536] XIA lock table entries: 1024 = 2^10 (4096 bytes)
[6914.969538] XIA loaded
[6919.581920] Hopscotch Hashing Table is Empty!!
[6919.581923] Find the location free_distance = 0, hop_info = 0!
[6919.581925] The new hop info = 1
[6919.581927] Add XID_TYPE 0x17, assigned virtual XIDTYPE 0
[6919.581929] Hopscotch Hashing Table is Empty!!
[6919.581931] Find the location free_distance = 0, hop_info = 1!
[6919.581932] The new hop info = 1
[6919.597666]
[6919.597666] **********Debug Info**********
[6919.597675] Bucket 23: key = 23, index = 0, hop_info = 1
[6919.597679] **********END**********
[6919.597679]
[6919.597691] The hash_index is 23

```
[ 6919.597694] XID Type : 0x17
[ 6919.597697] Index : 0
[ 6919.597700] Hop Info : 1
[ 6919.597705] The hash_index is 23
[ 6919.597708] XID Type : 0x17
[ 6919.597711] Index : 0
[ 6919.597714] Hop Info : 1
[ 6919.597719] The hash_index is 23
[ 6919.597722] XID Type : 0x17
[ 6919.597725] Index : 0
[ 6919.597728] Hop Info : 1
[ 6919.597732] The hash_index is 23
[ 6919.597735] XID Type : 0x17
[ 6919.597738] Index : 0
[ 6919.597741] Hop Info : 1
[ 6919.616377] The hash_index is 23
[ 6919.616381] XID Type : 0x17
[ 6919.616383] Index : 0
[ 6919.616384] Hop Info : 1
[ 6919.616477] XIA Principal XDP loaded
[ 6922.651000] Find the location free_distance = 0, hop_info = 0!
[ 6922.651003] The new hop info = 1
[ 6922.651005] Add XID_TYPE 0x16, assigned virtual XIDTYPE 1
[ 6922.651007] Find the location free_distance = 0, hop_info = 1!
[ 6922.651009] The new hop info = 1
[ 6922.664757]
[ 6922.664757] *******************Debug Info*******************
[ 6922.664763] Bucket 22: key = 22, index = 1, hop_info = 1
[ 6922.664766] Bucket 23: key = 23, index = 0, hop_info = 1
[ 6922.664768] **********************END**********************
[ 6922.664768]
[ 6922.664778] The hash_index is 22
[ 6922.664780] XID Type : 0x16
[ 6922.664782] Index : 1
[ 6922.664784] Hop Info : 1
[ 6922.664788] The hash_index is 22
[ 6922.664790] XID Type : 0x16
[ 6922.664792] Index : 1
[ 6922.664794] Hop Info : 1
[ 6922.664797] The hash_index is 22
[ 6922.664799] XID Type : 0x16
[ 6922.664801] Index : 1
[ 6922.664803] Hop Info : 1
[ 6922.664806] The hash_index is 22
```

```
[ 6922.664808] XID Type : 0x16
[ 6922.664810] Index : 1
[ 6922.664812] Hop Info : 1
[ 6922.679878] XIA Principal U4ID loaded
[ 7027.723730] The hash_index is 22
[ 7027.723733] XID Type : 0x16
[ 7027.723735] Index : 1
[ 7027.723737] Hop Info : 1
[ 7027.732467] The hash_index is 22
[ 7027.732472] XID Type : 0x16
[ 7027.732474] Index : 1
[ 7027.732477] Hop Info : 1
[ 7036.702101] The hash_index is 23
[ 7036.702105] XID Type : 0x17
[ 7036.702107] Index : 0
[ 7036.702109] Hop Info : 1
[ 7036.713335] The hash_index is 23
[ 7036.713339] XID Type : 0x17
[ 7036.713342] Index : 0
[ 7036.713344] Hop Info : 1
[ 7052.670492] The hash_index is 22
[ 7052.670496] XID Type : 0x16
[ 7052.670498] Index : 1
[ 7052.670499] Hop Info : 1
[ 7052.694800] The hash_index is 22
[ 7052.694804] XID Type : 0x16
[ 7052.694806] Index : 1
[ 7052.694808] Hop Info : 1
[ 7056.194508] The hash_index is 23
[ 7056.194512] XID Type : 0x17
[ 7056.194514] Index : 0
[ 7056.194515] Hop Info : 1
[ 7056.202736] The hash_index is 23
[ 7056.202740] XID Type : 0x17
[ 7056.202743] Index : 0
[ 7056.202745] Hop Info : 1
[ 7059.592206] The hash_index is 22
[ 7059.592210] XID Type : 0x16
[ 7059.592212] Index : 1
[ 7059.592214] Hop Info : 1
[ 7059.592225] The hash_index is 22
[ 7059.592228] XID Type : 0x16
[ 7059.592230] Index : 1
[ 7059.592232] Hop Info : 1
```

```
[ 7059.592285] The hash_index is 22
[ 7059.592287] XID Type : 0x16
[ 7059.592289] Index : 1
[ 7059.592291] Hop Info : 1
[ 7059.592294] The hash_index is 23
[ 7059.592295] XID Type : 0x17
[ 7059.592297] Index : 0
[ 7059.592299] Hop Info : 1
[ 7059.592418] The hash_index is 22
[ 7059.592422] XID Type : 0x16
[ 7059.592424] Index : 1
[ 7059.592426] Hop Info : 1
[ 7059.592432] The hash_index is 22
[ 7059.592433] XID Type : 0x16
[ 7059.592435] Index : 1
[ 7059.592437] Hop Info : 1
[ 7059.592466] The hash_index is 22
[ 7059.592468] XID Type : 0x16
[ 7059.592470] Index : 1
[ 7059.592472] Hop Info : 1
[ 7059.592474] The hash_index is 23
[ 7059.592476] XID Type : 0x17
[ 7059.592478] Index : 0
[ 7059.592480] Hop Info : 1
[ 7061.222328] The hash_index is 22
[ 7061.222333] XID Type : 0x16
[ 7061.222335] Index : 1
[ 7061.222337] Hop Info : 1
[ 7061.222423] The hash_index is 22
[ 7061.222425] XID Type : 0x16
[ 7061.222427] Index : 1
[ 7061.222429] Hop Info : 1
[ 7177.047860] The hash_index is 23
[ 7177.047864] XID Type : 0x17
[ 7177.047865] Index : 0
[ 7177.047867] Hop Info : 1
[ 7177.079701] The hash_index is 23
[ 7177.079705] XID Type : 0x17
[ 7177.079707] Index : 0
[ 7177.079709] Hop Info : 1
[ 7177.087736] The hash_index is 23
[ 7177.087739] XID Type : 0x17
[ 7177.087741] Index : 0
[ 7177.087743] Hop Info : 1
```

```
[ 7177.103794] The hash_index is 23
[ 7177.103798] XID Type : 0x17
[ 7177.103800] Index : 0
[ 7177.103802] Hop Info : 1
[ 7177.127720] The hash_index is 23
[ 7177.127722] XID Type : 0x17
[ 7177.127724] Index : 0
[ 7177.127726] Hop Info : 1
[ 7177.145167] To unload XID Type 0x17!
[ 7177.155789]
[ 7177.155789] *******************Debug Info*******************
[ 7177.155793] Bucket 22: key = 22, index = 1, hop_info = 1
[ 7177.155795] **********************END**********************
[ 7177.155795]
[ 7177.172645] XIA Principal XDP UNloaded
[ 7180.635728] The hash_index is 22
[ 7180.635734] XID Type : 0x16
[ 7180.635736] Index : 1
[ 7180.635738] Hop Info : 1
[ 7180.647739] The hash_index is 22
[ 7180.647743] XID Type : 0x16
[ 7180.647744] Index : 1
[ 7180.647746] Hop Info : 1
[ 7180.671770] The hash_index is 22
[ 7180.671773] XID Type : 0x16
[ 7180.671775] Index : 1
[ 7180.671776] Hop Info : 1
[ 7180.687858] The hash_index is 22
[ 7180.687862] XID Type : 0x16
[ 7180.687864] Index : 1
[ 7180.687866] Hop Info : 1
[ 7180.711738] To unload XID Type 0x16!
[ 7180.721026]
[ 7180.721026] *******************Debug Info*******************
[ 7180.721037] **********************END**********************
[ 7180.721037]
[ 7180.727822] XIA Principal U4ID UNloaded
[ 7183.739240] NET: Unregistered protocol family 41
[ 7186.272604] XIA UNloaded
```