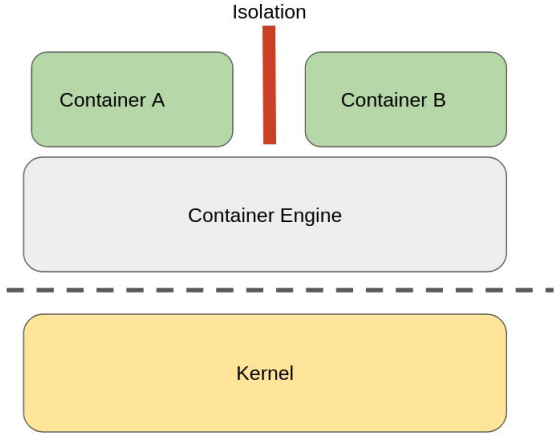
Karlsruher Institut für Technologie

Linux Namespaes

KIT: Testing OS-Level Virtualization for Functional Interference Bugs

Jean Diestl | 6. Juni 2023

Container


Isolation
Container A
Container B
Container Engine
Kernel

Namespaces

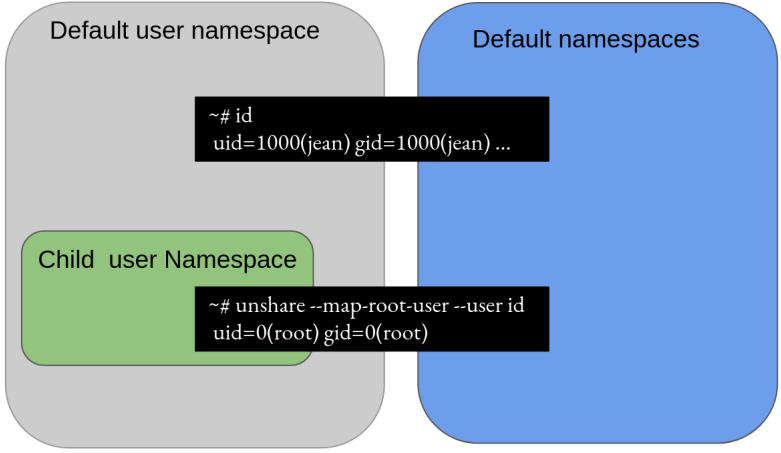
What are namespaces?

- process virtualization
- isolates global resources
- 7 types
- management via system calls
- **All namespaces share the same kernel**

Namespace types:

- Mount
- Process ID
- Network
- IPC
- User ID
- Control group
- UTS
- time namespace

Simple Example



```c
int child(){
    struct utsname uts;
    sethostname("container",9);
    uname(&uts);
    printf("[C] hostname: %s\n",uts.nodename);
    sleep(4);
    unshare(CLONE_NEWUTS);
    sethostname("container2",10);
    uname(&uts);
    printf("[C] new hostname %s\n",uts.nodename);
    return 0;
}
```

```
[P] parent hostname demo-host
[C] hostname: container
[P] child hostname container
[C] new hostname container2
[P] child hostname container
```

```c
int main(){
    unshare(CLONE_NEWUSER);
    int pid = clone(child,stack+4096,CLONE_NEWUTS|SIGCHLD,NULL);
    struct utsname uts;
    uname(&uts);
    printf("[P] parent hostname %s\n",uts.nodename);
    sleep(1);
    char path[100];
    snprintf(path,sizeof(path),"/proc/%d/ns/uts",pid);
    int fd = open(path,O_RDONLY);
    setns(fd,CLONE_NEWUTS);
    uname(&uts);
    printf("[P] child hostname %s\n",uts.nodename);
    waitpid(pid,0,0);
    uname(&uts);
    printf("[P] child hostname %s\n",uts.nodename);
    free(stack);
    return 0;
}
```

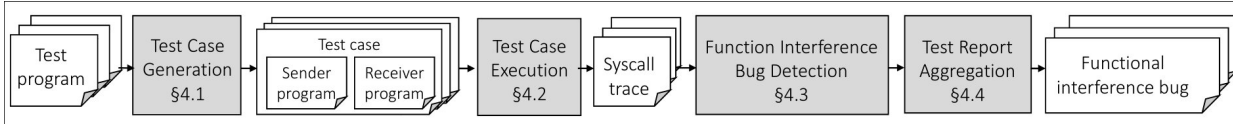
Interference bugs

- complex code base
- normal fuzzing does not work
  - hard to generate testcases that reach deep enough in kernel code
  - bugs might not be detected
  - many bug classes need multiple containers
- functional interference bugs

```
static int ptype_seq_show(...){
    ...
    if (pt->dev == NULL || dev_net(pt->dev)==seq_file_net(seq))
        if (pt->type == htons(ETH_P_ALL))
            seq_puts(seq,"ALL ");
        else
    }
    ...
    }
```

KIT
Kernel Isolation Tester

- focuses on functional interference
- implemented in around 7500 lines of code
- found 9 bugs
- pipeline design



Source: C. Liu,S. Gong and P. fonseca, "KIT Testing OS-Level Virtualization for Functional InterferenceBugs"(ASPLOS'23)

Test case generation

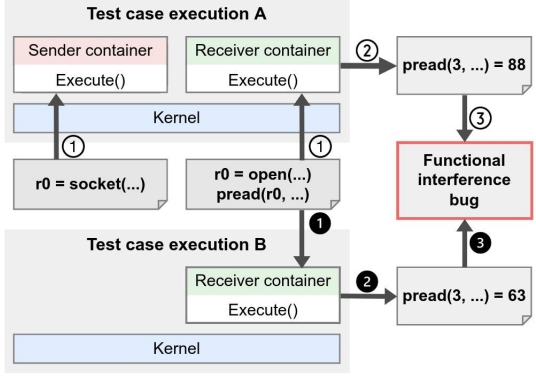
- take test program as input
- find shared memory access
- cluster test cases to reduce load

clustering

We only need to test one test case per cluster.

Test case execution

- system call traces and kernel stack traces are collected
- comapre sysetm call trees as AST
- limited to predictable systemcalls



Source: C. Liu,S. Gong and P. fonseca, "KIT Testing OS-Level Virtualization for Functional InterferenceBugs"(ASPLOS'23)

Test report aggregation

- identify system call pair that triggers bug
- group reports by triggering system call pair
- show only one report per group

Conclusion

- KIT is a dynamic kernel isolation test framework for system calls
- KIT seems to be effective
- KIT lacks support for system calls with dynamic output
- KIT only covers system call traces