Marek Maślanka

Let's meet the kernel livepatch feature

UBUNTU SUMMIT 2022

SOFTWARE MEETS HARDWARE

Semihalf

Who are we?

Marek Maślanka

- ARM/x86 embedded systems engineer @Semihalf
- Member of ChromeOS Kernel Team

Semihalf

- Low level, embedded software, hardware
- Linux & FreeBSD
- Networking: DPDK, ODP, VPP, SDN
- Bootloaders: U-Boot, EDK2, Coreboot
- Firmware: FreeRTOS, Zephyr, EC OS
- Virtualization
- Compilers
- FPGA: DMA, 400G SmartNIC
- EE

Copyright © 2022 Semihalf. All rights reserved.

Semihalf partners

Copyright © 2022 Semihalf. All rights reserved.

What is a Kernel Livepatch

- Fix security vulnerabilities without reboot
- Livepatch services
  - Canonical Livepatch Service for Ubuntu
  - Ksplice for Oracle Linux
  - SUSE Live Patching (previously known as Kgraft) for SUSE Enterprise Linux
  - kpatch for Red Hat Enterprise Linux
  - KernelCare for most major Linux distributions
- Livepatch as a kernel module
  - Fixes at functions level
  - Livetime until reboot

Copyright © 2022 Semihalf. All rights reserved.

How it works

- ftrace infrastructure
  - Compiler profiling feature
  - Insert *call* instruction to call *fentry*
- At kernel boot
  - Replace *call* to *nop* instruction
- On trace function
  - Replace *nop* to *call* to ftrace_caller

```
0x811eba10:      e8 db 5a a1 00      callq  0x81c014f0 <__fentry__>
0x811eba15:      55                  push   %rbp
0x811eba16:      53                  push   %rbx
0x811eba17:      48 89 fb            mov    %rdi,%rbx
0x811eba1a:      48 83 ef 80         sub    $0xffffff80,%rdi
0x811eba1e:      89 f5               mov    %esi,%ebp
0x811eba20:      e8 6b 6b 86 00      callq  0x81a52590 <_raw_spin_lock>
```

```
0x811eba10:      0f 1f 44 00 00      nop
0x811eba15:      55                  push   %rbp
0x811eba16:      53                  push   %rbx
0x811eba17:      48 89 fb            mov    %rdi,%rbx
0x811eba1a:      48 83 ef 80         sub    $0xffffff80,%rdi
0x811eba1e:      89 f5               mov    %esi,%ebp
0x811eba20:      e8 6b 6b 86 00      callq  0x81a52590 <_raw_spin_lock>
```

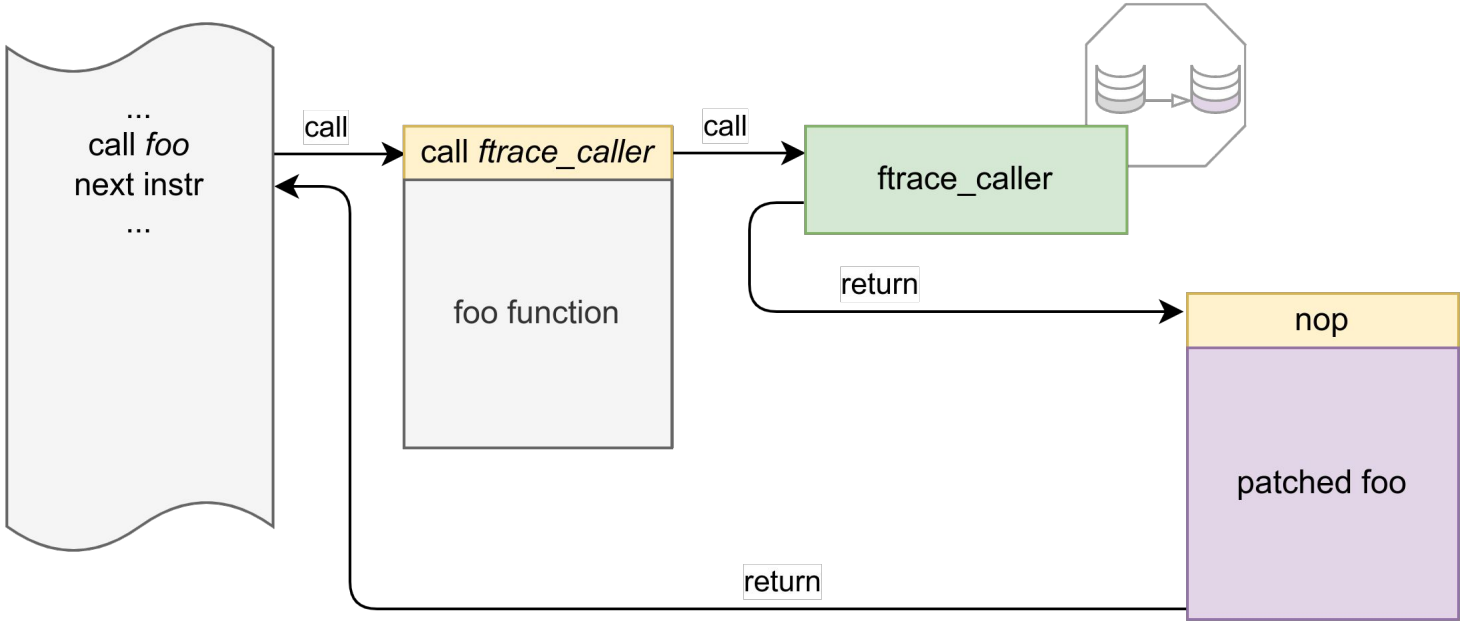
call ftrace_caller

```
0x811eba10:      e8 eb 45 e1 3e      callq  0xc0000000
0x811eba15:      55                  push   %rbp
0x811eba16:      53                  push   %rbx
0x811eba17:      48 89 fb            mov    %rdi,%rbx
0x811eba1a:      48 83 ef 80         sub    $0xffffff80,%rdi
0x811eba1e:      89 f5               mov    %esi,%ebp
0x811eba20:      e8 6b 6b 86 00      callq  0x81a52590 <_raw_spin_lock>
```


Copyright © 2022 Semihalf. All rights reserved.

Semihalf

How it works



Copyright © 2022 Semihalf. All rights reserved.

Semihalf

Prepare the patch

- Prerequisites
  - CONFIG_LIVEPATCH
    - CONFIG_FUNCTION_TRACER
    - CONFIG_KALLSYMS_ALL
  - Kernel headers

```
scripts/config -e FUNCTION_TRACER
scripts/config -e KALLSYMS_ALL
scripts/config -e LIVEPATCH
```


Copyright © 2022 Semihalf. All rights reserved.

Livepatch structures

```
struct klp_patch {
    struct module *mod;
    struct klp_object *objs;

    ...
};
```

```
struct klp_object {
    const char *name;
    struct klp_func *funcs;

    ...
};
```

```
struct klp_func {
    const char *old_name;
    void *new_func;

    ...
};
```

```
int klp_enable_patch(struct klp_patch *);

MODULE_INFO(livepatch, "Y");
```


Copyright © 2022 Semihalf. All rights reserved.

Example

- CVE-2021-22555
  - Buffer overflow
  - Gain Privilege

```
diff --git a/net/netfilter/x_tables.c b/net/netfilter/x_tables.c
index 6bd31a7a27fc5..92e9d4ebc5e8d 100644
--- a/net/netfilter/x_tables.c
+++ b/net/netfilter/x_tables.c
@@ -733,7 +733,7 @@ void xt_compat_match_from_user(struct xt_entry_match *m, void **dstptr,
 {
        const struct xt_match *match = m->u.kernel.match;
        struct compat_xt_entry_match *cm = (struct compat_xt_entry_match *)m;
-       int pad, off = xt_compat_match_offset(match);
+       int off = xt_compat_match_offset(match);
        u_int16_t msize = cm->u.user.match_size;
        char name[sizeof(m->u.user.name)];
@@ -743,9 +743,6 @@ void xt_compat_match_from_user(struct xt_entry_match *m, void **dstptr,
                match->compat_from_user(m->data, cm->data);
        else
                memcpy(m->data, cm->data, msize - sizeof(*cm));
-       pad = XT_ALIGN(match->matchsize) - match->matchsize;
-       if (pad > 0)
-               memset(m->data + match->matchsize, 0, pad);
        msize += off;
        m->u.user.match_size = msize;
@@ -1116,7 +1113,7 @@ void xt_compat_target_from_user(struct xt_entry_target *t, void **dstptr,
 {
        const struct xt_target *target = t->u.kernel.target;
        struct compat_xt_entry_target *ct = (struct compat_xt_entry_target *)t;
-       int pad, off = xt_compat_target_offset(target);
+       int off = xt_compat_target_offset(target);
        u_int16_t tsize = ct->u.user.target_size;
        char name[sizeof(t->u.user.name)];
@@ -1126,9 +1123,6 @@ void xt_compat_target_from_user(struct xt_entry_target *t, void **dstptr,
                target->compat_from_user(t->data, ct->data);
        else
                memcpy(t->data, ct->data, tsize - sizeof(*ct));
-       pad = XT_ALIGN(target->targetsize) - target->targetsize;
-       if (pad > 0)
-               memset(t->data + target->targetsize, 0, pad);
        tsize += off;
        t->u.user.target_size = tsize;
```


Copyright © 2022 Semihalf. All rights reserved.

Semihalf

Demo

Copyright © 2022 Semihalf. All rights reserved.

Example

- CVE-2021-31440
  - Buffer overflow
  - Gain Privilege

```diff
diff --git a/kernel/bpf/verifier.c b/kernel/bpf/verifier.c
index 637462e9b6ee9..9145f88b2a0a5 100644
--- a/kernel/bpf/verifier.c
+++ b/kernel/bpf/verifier.c
@@ -1398,9 +1398,7 @@ static bool __reg64_bound_s32(s64 a)
 static bool __reg64_bound_u32(u64 a)
 {
-        if (a > U32_MIN && a < U32_MAX)
-                return true;
-        return false;
+        return a > U32_MIN && a < U32_MAX;
 }
 static void __reg_combine_64_into_32(struct bpf_reg_state *reg)
@@ -1411,10 +1409,10 @@ static void __reg_combine_64_into_32(struct bpf_reg_state *reg)
                reg->s32_min_value = (s32)reg->smin_value;
                reg->s32_max_value = (s32)reg->smax_value;
        }
-        if (__reg64_bound_u32(reg->umin_value))
+        if (__reg64_bound_u32(reg->umin_value) && __reg64_bound_u32(reg->umax_value)) {
                reg->u32_min_value = (u32)reg->umin_value;
-        if (__reg64_bound_u32(reg->umax_value))
                reg->u32_max_value = (u32)reg->umax_value;
+        }
        /* Intersecting with the old var_off might have improved our bounds
         * slightly.  e.g. if umax was 0x7f...f and var_off was (0; 0xf...fc),
```


Copyright © 2022 Semihalf. All rights reserved.

Semihalf

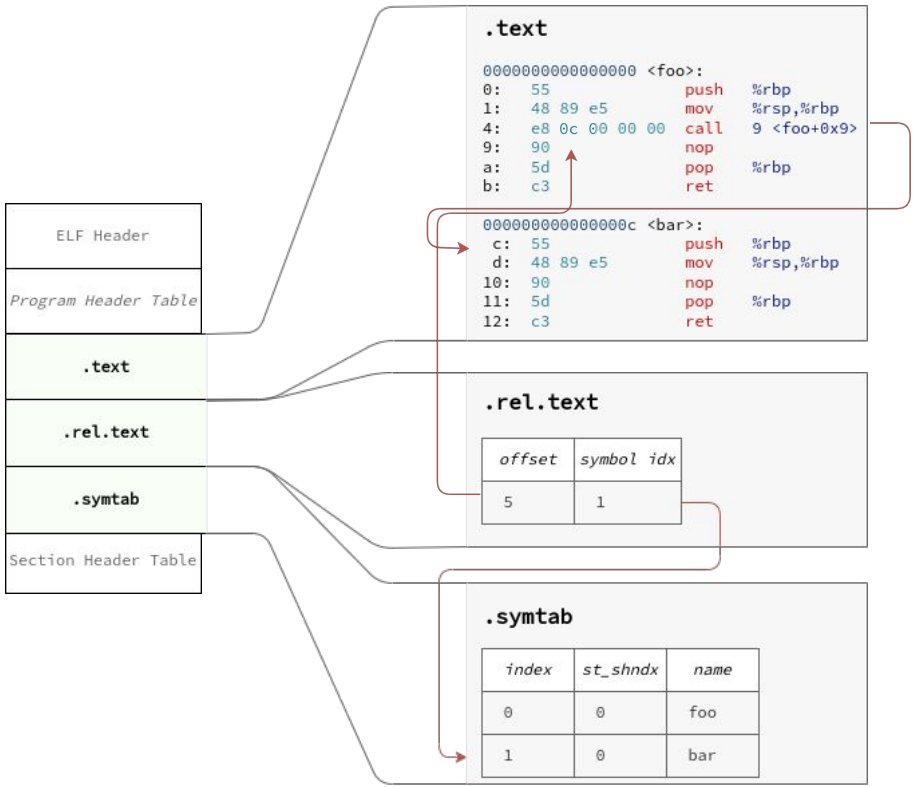
ELF Relocations

example.c

```c
void bar();

void foo()
{
    bar();
}

void bar()
{
}
```



Copyright © 2022 Semihalf. All rights reserved.

Semihalf

ELF Relocations

livepatch.c

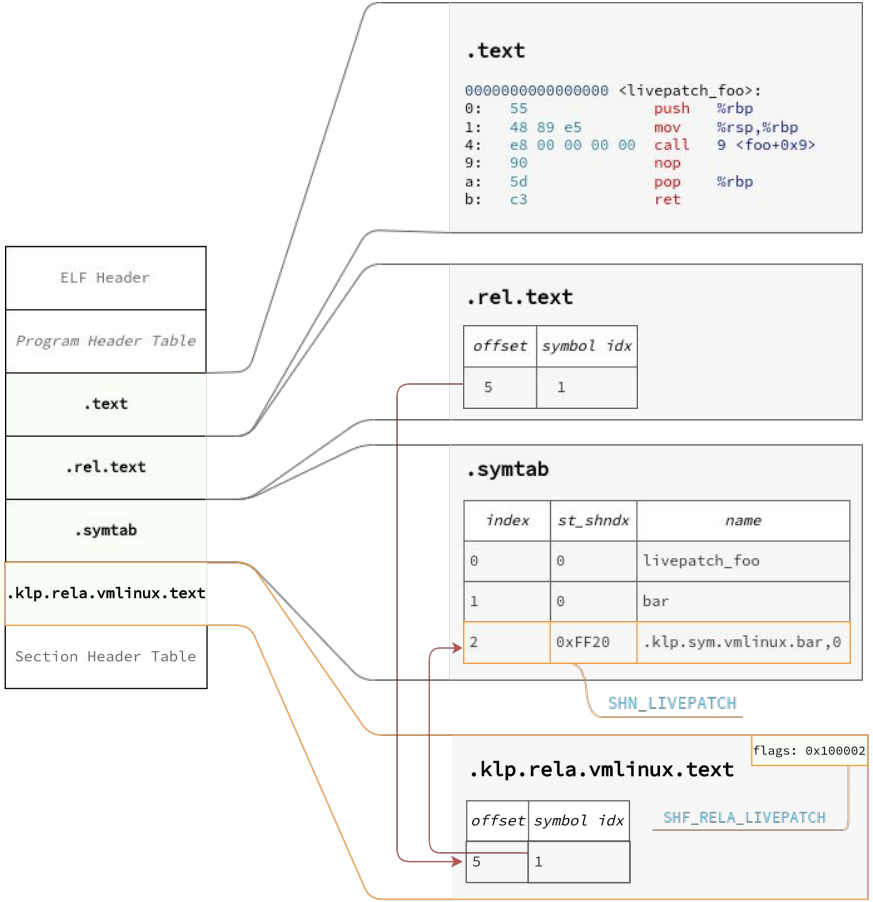
```
#include <linux/kernel.h>
#include <linux/module.h>

void livepatch_foo()
{
    bar();
}

MODULE_LICENSE("GPL");
```

Makefile

```
KBUILD_MODPOST_WARN = 1
```



Copyright © 2022 Semihalf. All rights reserved.

Semihalf

mklivepatch

- Part of the DEKU project
  - https://github.com/Semihalf/deku
- Build
  - $ *make*

Copyright © 2022 Semihalf. All rights reserved.

Semihalf

```
mklivepatch -s obj.function_to_patch -r obj.rel_fun,idx module.ko
```

- object where patching function lives
- function that need to be patched
- object where the function lives
- name of function that we need to get access
- index of a function if multiple functions exists with the same name in the object
- livepatching module file

Copyright © 2022 Semihalf. All rights reserved.

mklivepatch -s obj.function_to_patch -r obj.rel_fun,idx module.ko

```
mklivepatch -s vmlinux.foo -r vmlinux.bar,0 livepatch.ko
```


Copyright © 2022 Semihalf. All rights reserved.

Semihalf

Caveats

- Inline functions
  - /proc/kallsyms
  - readelf, objdump <file.o>
- Static variables
  - relocations for variables
- Global variables
  - variables as extern & modify relocations
- Compiler optimizations
  - Split functions
    - .cold.<N>, .part.<N>
  - Remove functions parameters

Copyright © 2022 Semihalf. All rights reserved.

Livepatch abilities

- Patch the data
  - modify static and globals variables
- Patch the structures
  - shadow variables
- Run -pre -post patch actions

Copyright © 2022 Semihalf. All rights reserved.

Semihalf

Tools

- DEKU (*https://github.com/Semihalf/deku*)
  - very fast
  - only small patches
  - requires kernel build artifacts
- kpatch (*https://github.com/dynup/kpatch*)
  - more mature project
  - hard to integrate with build systems

Copyright © 2022 Semihalf. All rights reserved.

Q&A