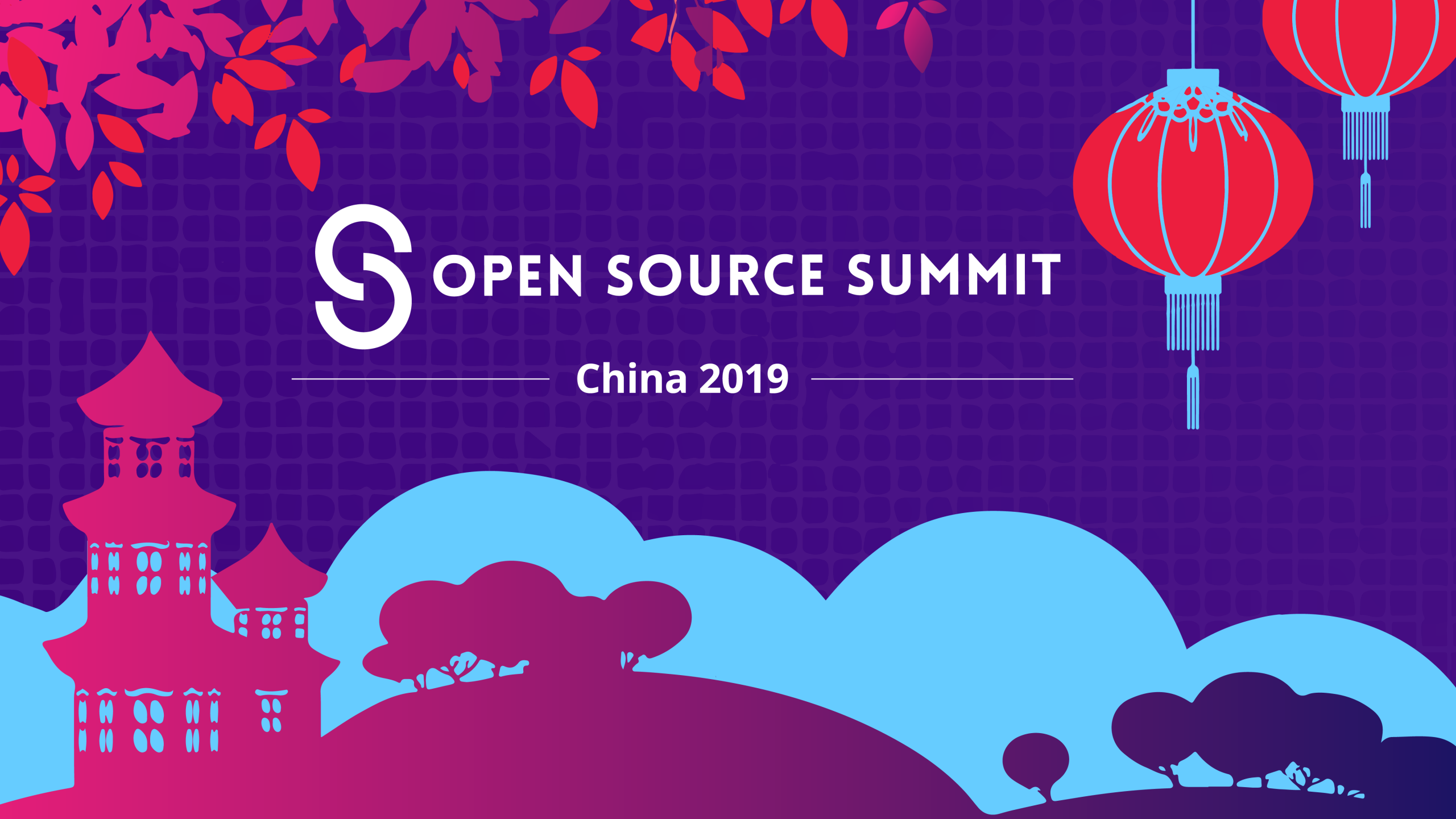

OPEN SOURCE SUMMIT
China 2019

Linux内核热补丁技术

严海双
<yanhaishuang@cmss.chinamobile.com>

什么是内核热补丁

- 内核在线热升级
  - 函数级别的在线替换
  - 打补丁过程无需重启系统
  - 不会影响线上应用
- 安全漏洞热修复
  - 只修复比较小的 BUG 改动
  - 无法适用于内核通用版本升级

为什么使用内核热补丁

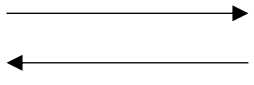
- 大量内核安全漏洞待修复
- 内核版本升级意味着需要重启
  - 影响线上运行的业务
  - 硬重启可能有失败的风险
- 重启会导致高成本的 downtime 时间
  - 有些应用无法接受超时100ms的中断
- 重新计划重启时间

从 Ftrace 到 Kpatch

Ftrace

- 内核函数跟踪
- GCC编译选项: -pg -fentry
- 增加fentry函数调用
  - 所有函数在入口处调用fentry
  - fentry作为跳板

```
<schedule>:
callq  ffffffff81a01d30 <__fentry__>
mov    %gs:0x15c40,%rax
push   %rbx
...
```



```
<__fentry__>:
retq
nopl   0x0(%rax,%rax,1)
nopw   %cs:0x0(%rax,%rax,1)
```

动态 Ftrace

- 无需直接调用fentrys
  - 开销太大
  - 性能影响接近13%
- 在系统启动阶段将fentry调用替换为nopl指令

```
<schedule>:
nopl   0x0(%rax,%rax,1) [FTRACE NOP]
mov    %gs:0x15c40,%rax
push   %rbx
…
```

开启 Tracing


```
<schedule>:
0xffffffffa77fc9a0 <schedule>:  nopl   0x0(%rax,%rax,1) [FTRACE NOP]
0xffffffffa77fc9a5 <schedule+5>:        mov    %gs:0x15c40,%rax
0xffffffffa77fc9ae <schedule+14>:       push   %rbx
0xffffffffa77fc9af <schedule+15>:       mov    0x10(%rax),%rdx
0xffffffffa77fc9b3 <schedule+19>:       test   %rdx,%rdx
0xffffffffa77fc9b6 <schedule+22>:       je     0xffffffffa77fc9c2 <schedule+34>
0xffffffffa77fc9b8 <schedule+24>:       cmpq   $0x0,0xb90(%rax)
0xffffffffa77fc9c0 <schedule+32>:       je     0xffffffffa77fc9db <schedule+59>
0xffffffffa77fc9c2 <schedule+34>:       mov    %gs:0x15c40,%rbx
0xffffffffa77fc9cb <schedule+43>:       xor    %edi,%edi
0xffffffffa77fc9cd <schedule+45>:       callq  0xffffffffa77fc130 <__schedule>
```

开启 Tracing


```
<schedule>:
0xffffffffa77fc9a0 <schedule>:  callq  0xffffffffa7a01f20 <ftrace_graph_caller>
0xffffffffa77fc9a5 <schedule+5>:        mov    %gs:0x15c40,%rax
0xffffffffa77fc9ae <schedule+14>:       push   %rbx
0xffffffffa77fc9af <schedule+15>:       mov    0x10(%rax),%rdx
0xffffffffa77fc9b3 <schedule+19>:       test   %rdx,%rdx
0xffffffffa77fc9b6 <schedule+22>:       je     0xffffffffa77fc9c2 <schedule+34>
0xffffffffa77fc9b8 <schedule+24>:       cmpq   $0x0,0xb90(%rax)
0xffffffffa77fc9c0 <schedule+32>:       je     0xffffffffa77fc9db <schedule+59>
0xffffffffa77fc9c2 <schedule+34>:       mov    %gs:0x15c40,%rbx
0xffffffffa77fc9cb <schedule+43>:       xor    %edi,%edi
0xffffffffa77fc9cd <schedule+45>:       callq  0xffffffffa77fc130 <__schedule>
```

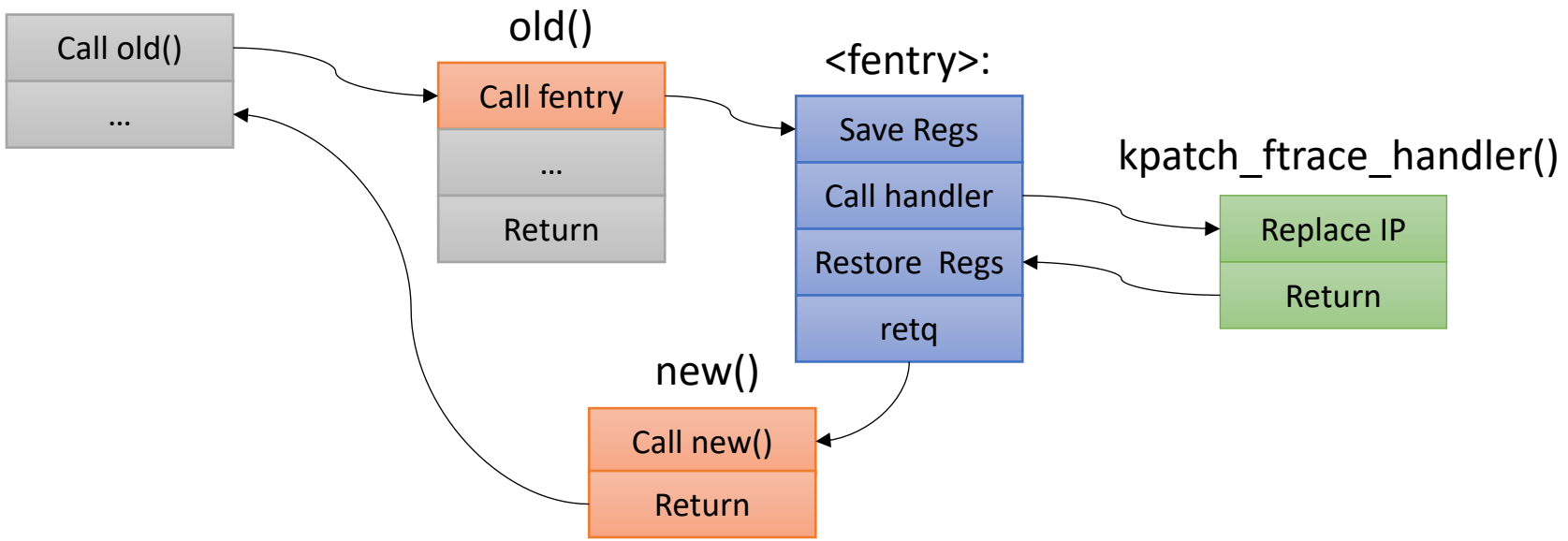
```
# echo schedule > set_ftrace_filter
# echo function_graph > current_tracer
# cat set_ftrace_filter
schedule
```

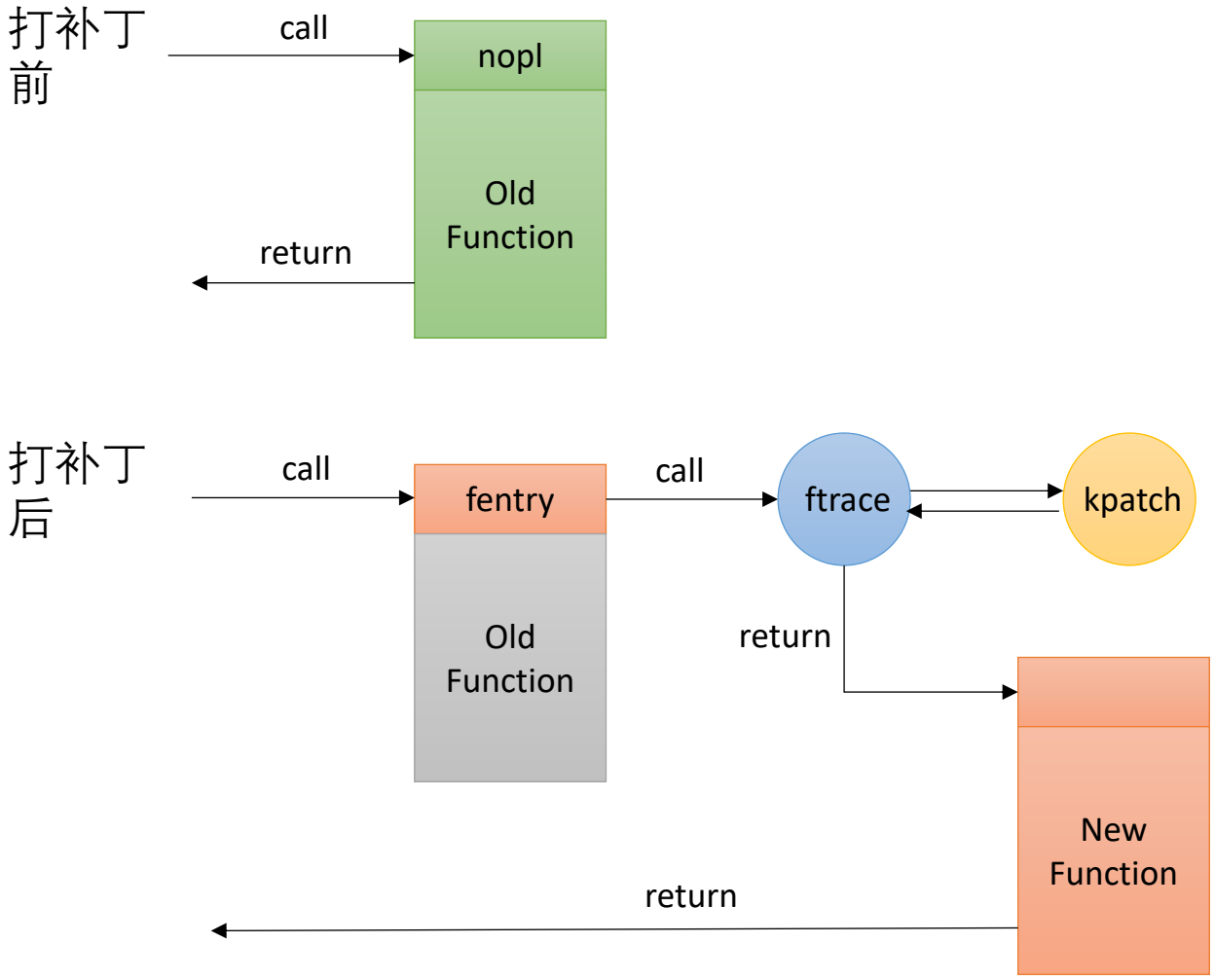
内核热补丁技术

Kpatch

- 增加 Ftrace 入口调用
- Stop_machine()
  - 停止所有正在运行进程一段时间
  - 禁用中断
- Safeness 检查
  - 遍历检查所有进程/线程调用栈
- 开启 hook 调用
  - 开启 Ftrace

- Kpatch 通过动态 Ftrace 技术打补丁
  - 在旧函数入口处开启 ftrace 并注册 handler
  - 替换 regs->ip 为新函数地址

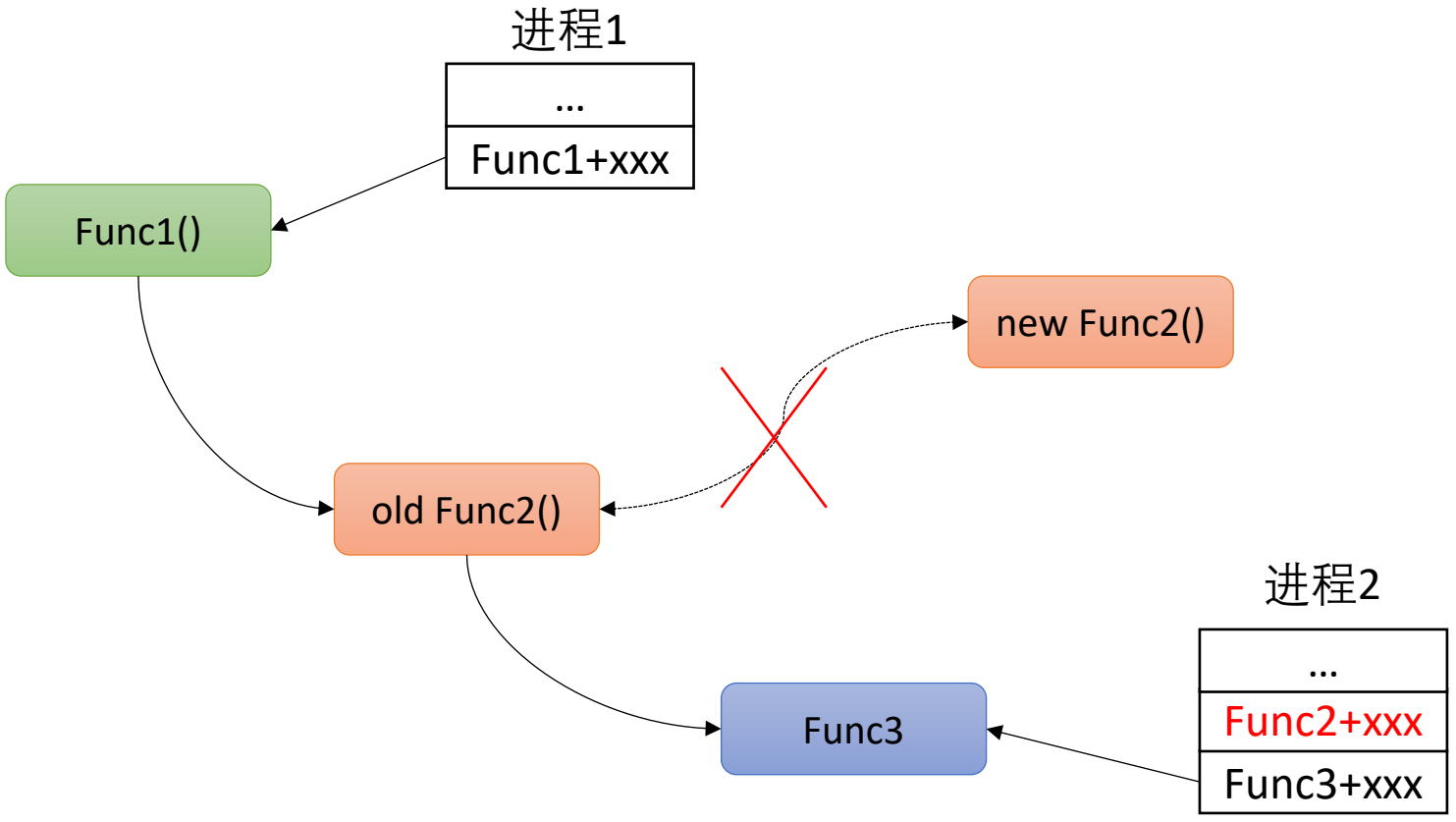



打补丁前
call
nopl
Old Function
return
打补丁后
call
fentry
call
ftrace
kpatch
Old Function
return
New Function
return

Activeness Safety 检查

- 验证 activeness safety
  - 确保所有进程当前调用栈没有包含要打上补丁的函数地址
  - 验证过程是在stop_machine()上下文中调用
  - 遍历所有线程调用栈，检查是否存在旧函数调用
  - 根据调用栈的回溯的地址信息进行验证

Activeness Safety 检查



Kpatch Tools

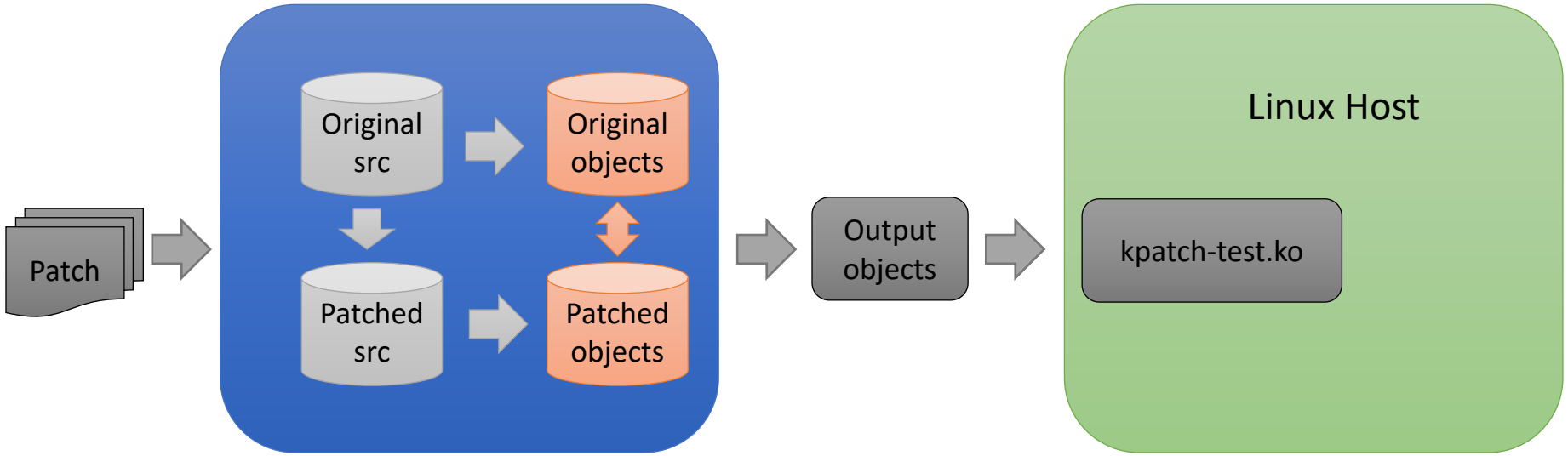
- kpatch-build
  - 将源码补丁文件转换为 Patch 模块的工具
- Patch 模块
  - 内核模块(.ko)，包含替换函数，原始函数的元数据信息
- Kpatch Core 模块
  - 内核模块(.ko)，主要提供接口给 Patch 模块注册替换函数

Kpatch Tools

- kpatch-build
  - 编译原始内核源码
    - 加上编译标记 -ffunction-sections -fdata-sections flags
  - 重新编译打上补丁的内核源码
    - 关注改动过的objects
  - 分析处理每个改动的objects
    - 生成目标objects
  - 链接所有生成的objects
    - 生成热补丁模块

Kpatch Tools

- 编译生成热补丁模块
  - kpatch-build test.patch -n kpatch-test
- 应用热补丁
  - kpatch load kpatch-test.ko



演示

应用限制

- 需要人为的进行安全评估！
- 带有 init 属性的函数无法应用热补丁
- 入口处没有 fentry 调用的函数无法应用热补丁
- 支持 80% 的 CVE 漏洞修复
- stop_machine() 导致延迟: 1ms-40ms


OPEN SOURCE SUMMIT
China 2019
谢谢