HotFix 热更新

2015-01-05 方栋
fangdong@baidu.com

Bai du 百度

概要

- 内核 Hotfix?
- 主流技术
- kpatch 原理
- kpatch 组件
  - kmod/core
  - kmod/patch
  - create-diff-object

内核 Hotfix？

- 内核在不停机的情况下进行安全升级

主流技术

- Ksplice
  - 商业软件，功能强大，支持新老内核版本 2008
- Kpatch
  - 开源，内核需要支持 ftrace 等 Redhat 2014
- Kgraft
  - 开源，同上 SUSE 2014

Kpatch 原理

- 对问题函数进行整体替换。通过用户态工具将 mod.patch 文件“编译”成内核模块，模块内使用 ftrace 机制来截获问题函数，并调用新函数

示例

```
$ kpatch-build -t vmlinux $bugfix.patch
Testing patch file
checking file fs/proc/meminfo.c
Building original kernel
Building patched kernel
Extracting new and modified ELF sections
meminfo.o: changed function: meminfo_proc_open
meminfo.o: changed function: meminfo_proc_show
Patched objects: vmlinux-3.13.0-32-generic
Building patch module: kpatch-mem_info.ko
```

Kpatch-build

- 编译 orig 内核
- 编译 patched 内核
- 遍历所有的 object 文件，使用 create-diff-object 工具生成 $target.o
- ld -r -o output.o $(find . -name "*.o")
- kmod/patch 与 output.o 生成内核模块

Kpatch 组件

- create-diff-object
  - diff 两个 object，找出需要升级的函数
- kmod/core
  - 核心内核模块，提供升级和卸载 API kpatch_register/kpatch_unregister
- kmod/patch
  - 内核模块，包含 hotfix 所需要的 meta 数据

kmod/core

- 通过 ftrace 截获函数调用，跳转到新的函数地址
- Safe-point
  - 函数不能出现在内核栈里， schedule()

kmod/core - Hotfix 过程

- 全局的 patched 函数链表
- 注册 ftrace 钩子，指令地址 ip
  - 检查链表，存在当前 ip 对应的 patched 函数，则跳转到新 ip
- stop_machine(apply_patch，data) 停止所有 cpu，让一个 cpu 单独执行 apply_patch 函数

kmod/core - Hotfix 过程

- 检查内核栈是否有活跃的 patched 函数
  - 活跃： hotfix 失败
  - 不活跃：挂载到全局的 patched 函数链表
- Hotfix 结束

kmod/patch

- 定义与 create-diff-object 工具的接口
  - patch_* 函数结构的定义
  - 等等

```
extern struct kpatch_patch_func __kpatch_funcs[], __kpatch_funcs_
    end[];
struct kpatch_patch_func {
    unsigned long new_addr;     // 新函数地址
    unsigned long new_size;     // 新函数长度
    unsigned long old_addr;     // 老函数地址
    unsigned long old_size;     // 老函数长度
    char *name;                 // 函数名
    char *objname;              // 模块名
};
```

kmod/patch

- 所有的 kpatch_patch_func 等 extern 数据都是 create-diff-object 写到 ELF 文件的 section 里的， kmod/patch 编译时需要依赖连接脚本 lds 进行处理

```
__kpatch_funcs = ADDR(.kpatch.funcs);
```

kmod/patch

- 调用 kpatch_register() 进行 hotfix

create-diff-object

- 内核编译需要依赖 -ffunction-sections -fdata-sections 参数，每个函数 or 变量等都会编译到独立的 section
- diff 两个 object 文件，找出需要升级的函数，找出不同的 section 和 symbol

create-diff-object

- 构造两个特殊的 section
  - kpatch.funcs 包含函数升级所需要的元数据，例如函数地址，大小等等
  - kpatch.dynmics
- 为上述 section 构造相应的重定位表
  - kmod/patch 通过 p_func->new_addr 可直接获得新的函数地址

create-diff-object

- 并重新生成一个 ELF 文件

Q&A