JudgePenguin

基于Linux的应用程序稳态测试系统

第九周进展

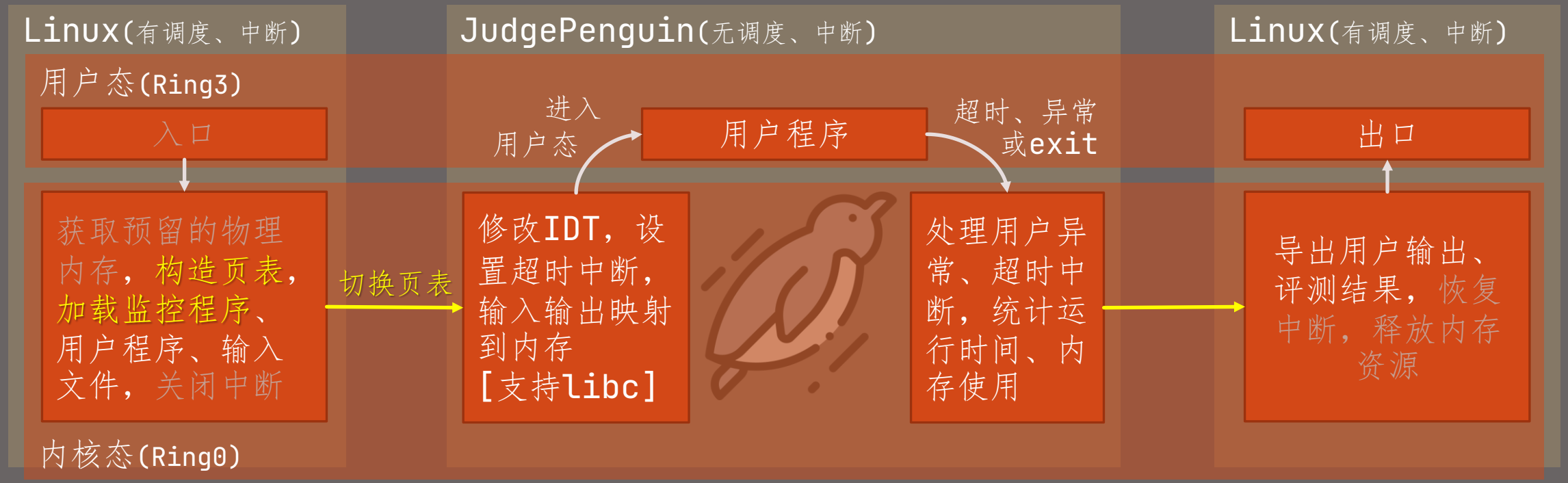
致理-信计01 单敬博 2020012711

2023/4/23

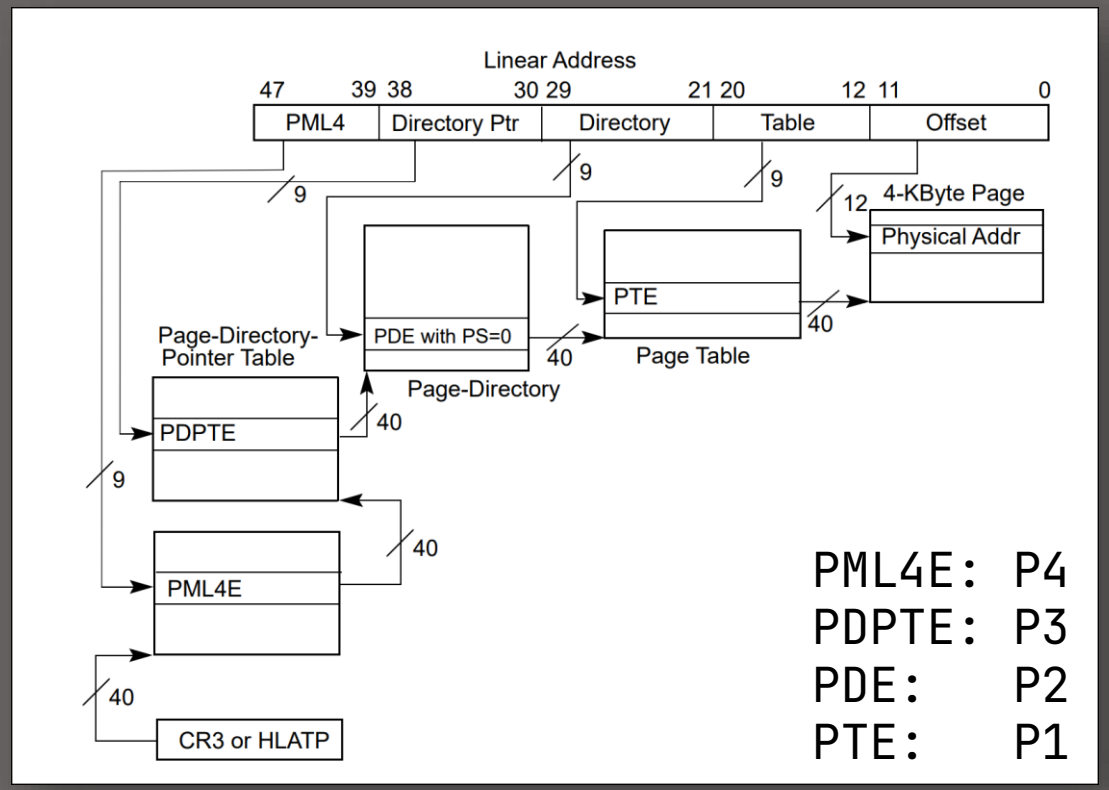
（上周的）下周计划

- 在预留内存中创建x86_64多级页表
- 设置跳板页，实现页表切换
- 参考JudgeDuck，调研监控程序的实现方式
  - 重点研究trap_handler、IDT配置、超时中断

本周进展


Linux(有调度、中断)
JudgePenguin(无调度、中断)
Linux(有调度、中断)
用户态(Ring3)
入口
进入用户态
用户程序
超时、异常或exit
出口
获取预留的物理内存，构造页表，加载监控程序、用户程序、输入文件，关闭中断
切换页表
修改IDT，设置超时中断，输入输出映射到内存[支持libc]
处理用户异常、超时中断，统计运行时间、内存使用
导出用户输出、评测结果，恢复中断，释放内存资源
内核态(Ring0)

x86_64四级页表



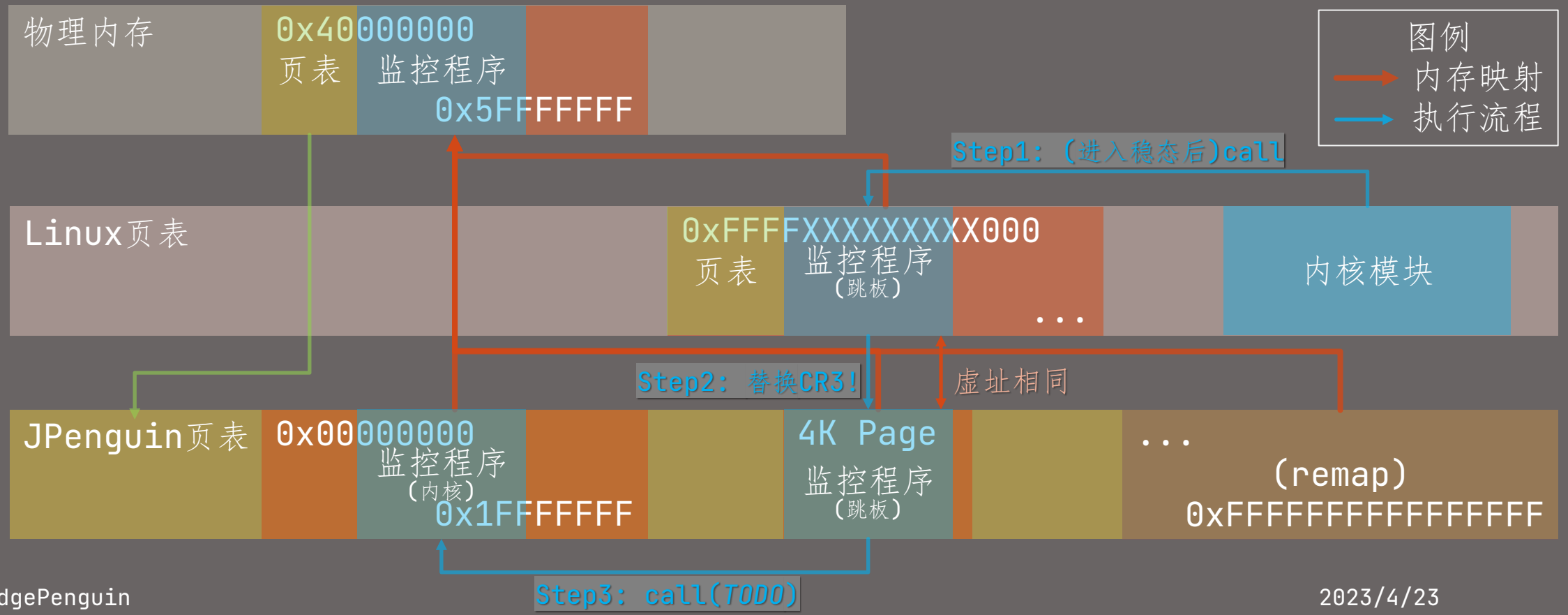
63	62	61	60	59	58	57	56	55	54	53	52	51	M[1]	M-1	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved[2]														Address of PML4 table (4-level paging) or PML5 table (5-level paging)																								Ignored						PCD	PWT	Ign.		CR3
XD[3]	Ignored												Rsvd.	Address of PML4 table																								R[4]	Ign.			Rsvd	Ign	A	PCD	PWT	U/S	R/W	1	PML5E: present
Ignored																																																0	PML5E: not present
XD	Ignored												Rsvd.	Address of page-directory-pointer table																								R	Ign.			Rsvd	Ign	A	PCD	PWT	U/S	R/W	1	PML4E: present
Ignored																																																0	PML4E: not present
XD	Prot. Key[5]		Ignored										Rsvd.	Address of 1GB page frame																								Reserved						PAT	R	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	PDPTE: 1GB page
XD	Ignored												Rsvd.	Address of page directory																								R	Ign.			0	Ign	A	PCD	PWT	U/S	R/W	1	PDPTE: page directory
Ignored																																																0	PDTPE: not present
XD	Prot. Key		Ignored										Rsvd.	Address of 2MB page frame																								Reserved						PAT	R	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Ignored												Rsvd.	Address of page table																								R	Ign.			0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																																																0	PDE: not present
XD	Prot. Key		Ignored										Rsvd.	Address of 4KB page frame																								R	Ign.			PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																																																0	PTE: not present

每级对应9位地址　每块4 KiB　每项64 bit

创建页表

- 使用预留内存的头部创建页表
- 使用P3 Page(1 GiB)将全部物理内存remap到虚存高地址(0x-1结束)
- 使用4K Page将预留内存映射到虚存低地址(0x0开始)
- 映射结束后计算page_table_break(在它后面加载监控程序)
- 对监控程序对应的物理内存(有限个Page)建立与Linux相同的虚存

页表切换


物理内存
0x40000000
页表
监控程序
0x5FFFFFFF
图例
内存映射
执行流程
Step1: (进入稳态后)call
Linux页表
0xFFFFXXXXXXXXX000
页表
监控程序
(跳板)
...
内核模块
Step2: 替换CR3!
虚址相同
JPenguin页表
0x00000000
监控程序
(内核)
0x1FFFFFFF
4K Page
监控程序
(跳板)
...
(remap)
0xFFFFFFFFFFFFFFFF
Step3: call(TODO)

加载监控程序

- 使用内核函数filp_open打开监控程序bin文件
- 使用内核函数kernel_read直接将bin读取到预留内存

- *TODO*: 学习jailhouse添加firmware文件的方法
  - 似乎需要向Linux注册一个“设备”

```
MODULE_FIRMWARE("kernel.bin");
```

简单的监控程序

- 在bin头部添加一个header
  - 保存签名(魔法字符串)、入口函数偏移
  - 提供内核模块与监控程序之间的[临时信道]

```
struct judge_penguin_header __attribute__((section(".header")))
header = {.signature = "JPenguin", .entry = entry, .magic = 0xdeadbeef};
```

```
--- a/hypervisor/include/jailhouse/header.h
+++ b/hypervisor/include/jailhouse/header.h
@@ -12,7 +12,7 @@

 #include <asm/jailhouse_header.h>

-#define JAILHOUSE_SIGNATURE    "JAILHOUS"
+#define JAILHOUSE_SIGNATURE    "RVMIMAGE"

 #define HYP_STUB_ABI_LEGACY 0
 #define HYP_STUB_ABI_OPCODE 1
```


- 保存环境 & 切换页表 & 恢复环境 (学习jailhouse)
  - 保存GDT TSS {C,D,E,F,G}S IDT CR{0,3,4}
  - 替换GDT CS CR3; 清空DS ES SS

简单的监控程序 cont.

- 好像还差点东西：给监控程序设置一个独立的内核栈
- 解决方案 （部分参考jailhouse）
  - 在bin中预留一段.data空间作为内核栈

    TODO：改为加载监控程序时分配，通过参数传递给监控程序
  - 在开始保存其他环境之前，保存rsp并切换到独立内核栈(汇编实现)
- 恢复环境大致就是以上过程取逆

切换页表之后

- 改完CR3之后，没爆炸(之前经历了数次死机/黑屏重启 XD)
- 但是如何确认页表替换成功了？
- 向一段只有JPenguin页表中才有效的内存写入一些东西

```
char *p = (char *)0x109024ull; // magic address in JPenguin mode
char *q = "Hello, world!";
for (int i = 0; i < 13; i++) {
  p[i] = q[i];
}
p[13] = '\0';
```

真机演示

```
kernel: initializing page table...
kernel: page table uses 265 4K blocks.
kernel: page table break set to [p]0x40109000.
kernel: P4: [p]0x40000000.
kernel: P3[upp]: [p]0x40001000.
kernel: P3[low]: [p]0x40002000.
kernel: P2[0]: [p]0x40003000.
kernel: page table initialized.
kernel: JudgePenguin: main begin
kernel: magic: deadbeef
kernel: entry_addr: [v]0xffffb750601097b2
kernel: kernel return: 0
kernel: magic: deadbef1
kernel: output[0] = 60109580
kernel: output[1] = ffffb750
kernel: output[2] = bb39a000
kernel: output[3] = 00000001
kernel: message from kernel: Hello, world!
kernel: JudgePenguin: main end
```

```
75060109000.
358850
47481773
706
36293175
```

```
kernel: disable interrupt on cpu 0.
kernel: disable interrupt on cpu 2.
kernel: disable interrupt on cpu 1.
kernel: disable interrupt on cpu 3.
kernel: test_rdtsc on cpu 0 for 100000000 rounds.
kernel: avg=21 max=100 bad_count=0.
kernel: test_rdtsc pass.
kernel: test_rdtsc on cpu 0 for 100000000 rounds.
kernel: avg=21 max=100 bad_count=0.
kernel: test_rdtsc pass.
kernel: test_rdtsc on cpu 0 for 100000000 rounds.
kernel: avg=21 max=100 bad_count=0.
kernel: test_rdtsc pass.
kernel: test_rdtsc on cpu 0 for 100000000 rounds.
kernel: avg=21 max=100 bad_count=0.
kernel: test_rdtsc pass.
kernel: test_rdtsc on cpu 0 for 100000000 rounds.
kernel: avg=21 max=98 bad_count=0.
kernel: test_rdtsc pass.
```

下周计划 ~~一周也许不能完全搞定~~

- 内核模块：
  - 研究向Linux添加设备，使用MODULE_FIRMWARE；
  - 构造监控程序内核栈
- 监控程序：分为跳板、内核两部分
  - 跳板部分：主要参考jailhouse，使用C、汇编
  - 内核部分：主要参考JudgeDuck，考虑使用rust?(rcore-os)
    - 配置IDT，设置超时中断(保存与还原8259A / LAPIC，可能较困难)
  - 调试方法：可能需要一台带串口的x86_64设备(或者QEMU)

感谢聆听 & 欢迎提问

致理-信计01 单敬博 2020012711