方法一：首先直接在dom0用户空间声明vmxon，vmxoff，vmread，vmwrite等函数，然后main函数直接调用。

结果：出现段错误，原因为权限不够。

方法二：创建一个Hypercall，利用 Privcmd从用户空间调用这个Hypercall

**1：首先注册一个hypercall**

在  xen/arch/x86/x86\_64/entry.S , make new lines under ENTRY(hypercall\_table) and ENTRY(hypercall\_args\_table).

ENTRY(hypercall\_table)

.quad do\_set\_trap\_tabble  
.quad do\_mmu\_update  
…  
.quad do\_tmem\_op  
.quad do\_vmx\_test   
.rept \_\_HYPERVISOR\_arch\_0-((.-hypercall\_table)/8)  
.quad do\_ni\_hypercall  
.endr  
.quad do\_mca  
.rept NR\_hypercalls-((.-hypercall\_table)/8)  
.endr

ENTRY(hypercall\_args\_table)

.byte 1 /\* do\_set\_trap\_table \*/  
.byte 4 /\* do\_mmu\_update \*/  
…  
.byte 1 /\* do\_tmem\_op \*/  
.byte 1 /\* do\_vmx\_test  
.rept \_\_HYPERVISOR\_arch\_0-(.-hypercall\_args\_table)  
.byte 0 /\* do\_ni\_hypercall \*/  
.endr  
.byte 1 /\* do\_mca \*/  
.rept NR\_hypercalls-(.-hypercall\_args\_table)  
.byte 0 /\* do\_ni\_hypercall \*/  
.endr

**定义Hypercall号：**

在  xen/include/public/xen.h ,

#define \_\_HYPERVISOR\_set\_trap\_table 0  
#define \_\_HYPERVISOR\_mmu\_update     1  
…  
#define \_\_HYPERVISOR\_tmem\_op        38  
#define \_\_HYPERVISOR\_jguno\_rdmsr     39  // new hypercall

**声明Hypercall：**

在 xen/include/xen/hypercall.h,

extern long  
do\_vmx\_test(  
int);

**HyperCall函数定义：**

为了获取方便，便于执行vmxoff、vmxon、vmread、vmwrite等测试，把函数定义放在了 xen/arch/x86/hvm/vmx/vmcs.c,如下：

long do\_vmx\_test(  
int input){  
printk(“vmx test hypercall called!\n”);  
switch(input)

case 1:

……………………….

Case 2:

………………………..

Case 3:

………………………  
return 1;  
}

**2. 利用xen提供的 Privcmd 驱动程序调用Hypercall**

**Define a Do Function Which Declares the Hypercall and Calls It.**

在 tools/libxc/***xc\_private.h*** 中do\_sysctl函数下加入以下调用函数：

static inline int do\_vmx\_test\_hypercall(xc\_interface \*xch, int input\_number)  
{  
int ret;  
DECLARE\_HYPERCALL;  
PERROR(“\*\*do\_vmx\_test\_hypercall\_buffer\* START\n”);

hypercall.op = \_\_HYPERVISOR\_vmx\_test\_rdmsr;//39  
hypercall.arg[0] = input\_number;

ret = do\_xen\_hypercall(xch, &hypercall); //利用Privcmd接口

PERROR(“\*\*do\_vmx\_test\_hypercall\_buffer\* STOP\n”);  
return ret;  
}

**添加一个函数***xc\_vmx\_test.c* **去调用：**

在 tools/xcutils/下加入这个c文件：.

#include <err.h>  
#include <stdlib.h>  
#include <stdint.h>  
#include <stdio.h>

#include <xenctrl.h>  
#include <xenguest.h>  
#include <xc\_private.h>

int  
main(int argc, char \*\*argv)  
{  
xc\_interface \*xch;  
int ret, input\_number;

if ( (argc != 2) )  
errx(1, “usage: %s input\_number “, argv[0]);

xch = xc\_interface\_open(0,0,0);  
if ( !xch ){  
errx(1, “xcutils: xc\_vmx\_test.c: failed to open control interface”);  
}

input\_number= atoi(argv[1]);  
printf(“input\_number: %d\n”, input\_number);

ret = do\_vmx\_test\_hypercall(xch, input\_number);  
printf(“return value: %d\n”, ret);

if ( ret == 0 )  
{  
errx(1, “ret == 0\n”);  
fflush(stdout);  
}

xc\_interface\_close(xch);

return ret;  
}

修改这个文件夹下的Makefile去编译这个文件：

#  
# tools/xcutils/Makefile  
#  
# This file is subject to the terms and conditions of the GNU General  
# Public License. See the file “COPYING” in the main directory of  
# this archive for more details.  
#  
# Copyright (C) 2005 by Christian Limpach  
#

XEN\_ROOT = $(CURDIR)/../..  
include $(XEN\_ROOT)/tools/Rules.mk

PROGRAMS = xc\_restore xc\_save readnotes lsevtchn xc\_vmx\_test

CFLAGS += -Werror

CFLAGS\_xc\_restore.o := $(CFLAGS\_libxenctrl) $(CFLAGS\_libxenguest)  
CFLAGS\_xc\_save.o := $(CFLAGS\_libxenctrl) $(CFLAGS\_libxenguest) $(CFLAGS\_libxenstore)  
CFLAGS\_readnotes.o := $(CFLAGS\_libxenctrl) $(CFLAGS\_libxenguest)  
CFLAGS\_lsevtchn.o := $(CFLAGS\_libxenctrl)  
CFLAGS\_xc\_vmx\_test.o := $(CFLAGS\_libxenctrl) $(CFLAGS\_libxenguest)

.PHONY: all  
all: build

.PHONY: build  
build: $(PROGRAMS)

xc\_restore: xc\_restore.o  
$(CC) $(LDFLAGS) $^ -o $@ $(LDLIBS\_libxenctrl) $(LDLIBS\_libxenguest) $(APPEND\_LDFLAGS)

xc\_save: xc\_save.o  
$(CC) $(LDFLAGS) $^ -o $@ $(LDLIBS\_libxenctrl) $(LDLIBS\_libxenguest) $(LDLIBS\_libxenstore) $(APPEND\_LDFLAGS)

readnotes: readnotes.o  
$(CC) $(LDFLAGS) $^ -o $@ $(LDLIBS\_libxenctrl) $(LDLIBS\_libxenguest) $(APPEND\_LDFLAGS)

lsevtchn: lsevtchn.o  
$(CC) $(LDFLAGS) $^ -o $@ $(LDLIBS\_libxenctrl) $(APPEND\_LDFLAGS)

xc\_vmx\_test: xc\_vmx\_test.o  
$(CC) $(LDFLAGS) $^ -o $@ $(LDLIBS\_libxenctrl) $(LDLIBS\_libxenguest) $(APPEND\_LDFLAGS)

*编译测试：*

**3. 函数测试：**

执行: xc\_vmx\_test

$ xc\_vmx\_test 1

测试vmread，vmwrite：

input\_number: 1  
xc: error: \*\*do\_vmx\_\_test\_hypercall\_buffer\* START  
(0 = Success): Internal error  
xc: error: \*\*do\_vmx\_test\_hypercall\_buffer\* STOP  
(0 = Success): Internal error  
return value: 1

利用xl dmesg擦看输出：.

(XEN) DO(vmx\_test) hypercall called!  
(XEN) vmread

(XEN) vmwrite

同样的，利用$ xc\_vmx\_test 2可以测试vmxoff。

利用$ xc\_vmx\_test 3可以测试vmxon。

分析：

（1）若在虚拟机运行过程中直接执行vmread，vmwrite，会引起电脑重启，只有在执行VMExit进入root模式才可安全执行vmread，vmwrite读取或写入vmcs结构相关属性值。

（2）默认情况下进入xen运行时VMXON开启vmx模式的，在创建虚拟机前可以执行VMXOFF关闭vmx模式；而当创建一个虚拟机后执行vmxoff会引起电脑重启。