OS systems Dry Part

HW 4

Submitters :

Shaked Or 308026400

Aviad Rozenkof 316592922

**Q1**

1.

int double\_wait (int fd1, int fd2){

//initialize a poolfd array called fds

struct pollfd fds[2];

//put the fdnums in the file descriptor field

(fds[0]).fd = fd1;

(fds[1]).fd = fd2;

//put the POLLIN num in the events field

(fds[0]).events = POLLIN;

(fds[1]).events = POLLIN;

return poll (fds, 2, -1);

}

2.

This function won't work because SIGUSR1 is still blocked in the beginning of this function.

3.

Imagine the following scenario:

After unblocking SIGUSR1 and before calling poll, SIGUSR1 is signaled.

In this case poll will start after the handler for SIGUSR1 returned. This is problematic because in this case xpoll returns only when SIGUSR1 was signaled and after 1 fd is ready to read (instead of or).

4.

(In this cause we use ppoll thereby making the polling and unmasking atomic)

int double\_wait\_safe(int fd1, int fd2){

//initialize new sig set with all but SIGUSR1

sigset\_t newmask;

sigfillset(&newmask);

sigdelset(&newmask,SIGUSR1);

//initialize a poolfd array called fds

struct pollfd fds\_arr[2], \*fds = fds\_arr;

//put the fdnums in the file descriptor field

(fds\_arr[0]).fds = fd1;

(fds\_arr[1]).fds = fd2;

//put the POLLIN num in the events field

(fds\_arr[0]).events = POLLIN;

(fds\_arr[1]).events = POLLIN;

//call poll

return ppoll (fds, 2, -1, &newmask);

}

**Q2**

1.

An adversary user could potentially call RNDCLEARPOOL over and over again, thereby jamming the processes of other users who are waiting for random data.

2.

A.

#include <sched.h>

void make\_me\_FIFO(){

//make sched\_param with MAXPRIO

struct sched\_param param;

param.sched\_priority=0;

// call setsched

sched\_setscheduler(0,SCHED\_FIFO,&param);

}

B.

#include "/dev/srandom"

#include <sched.h>

void make\_me\_FIFO(){

// load the faulty module

int status;

status=init\_module();

if(status!=0){

return -1;

}

struct sched\_param param;

param.sched\_priority=0;

//run set\_sched until it succeeds

while(sched\_setscheduler(0,SCHED\_FIFO,&param)!=0){

// mix ADDR for chance of getting bit 23 up

read(NULL,\_\_ADDR\_\_,4,NULL);

}

}

C.

The probability is which comes out to

=

D.

The expected value is 1/p =2