level0

태그	Assembly Shell
상태	완료

풀이과정

겪었던 어려움

풀이과정

gdb

시작하면 level0이라는 파일이 존재한다. 이 파일을 인자 없이 실행시키면 아래와 같은 에러가 뜬다.

```
level0@RainFall:~$ ./level0
Segmentation fault (core dumped)
```

여기서 `core dumped` 에러는 입력값이 잘못되었을 때 생기는 에러이다. 따라서 입력값을 아무거나 주고 실행해보면 에러 메시지가 아래와 같이 바뀌게 된다.

```
level0@RainFall:~$ ./level0 test
No !
```

그 이후 수확이 없어서 gdb를 통해서 disassmbly를 통해서 main부분을 확인해봤다.

```
(gdb) disas main
Dump of assembler code for function main:
   0x08048ec0 <+0>:	push   %ebp
   0x08048ec1 <+1>:	mov    %esp,%ebp
   0x08048ec3 <+3>:	and    $0xfffffff0,%esp
   0x08048ec6 <+6>:	sub    $0x20,%esp
   0x08048ec9 <+9>:	mov    0xc(%ebp),%eax
```

```
   0x08048ecc <+12>:    add    $0x4,%eax
   0x08048ecf <+15>:    mov    (%eax),%eax
   0x08048ed1 <+17>:    mov    %eax,(%esp)
   0x08048ed4 <+20>:    call   0x8049710 <atoi>
   0x08048ed9 <+25>:    cmp    $0x1a7,%eax
   0x08048ede <+30>:    jne    0x8048f58 <main+152>
   0x08048ee0 <+32>:    movl   $0x80c5348,(%esp)
   0x08048ee7 <+39>:    call   0x8050bf0 <strdup>
   0x08048eec <+44>:    mov    %eax,0x10(%esp)
   0x08048ef0 <+48>:    movl   $0x0,0x14(%esp)
   0x08048ef8 <+56>:    call   0x8054680 <getegid>
   0x08048efd <+61>:    mov    %eax,0x1c(%esp)
   0x08048f01 <+65>:    call   0x8054670 <geteuid>
   0x08048f06 <+70>:    mov    %eax,0x18(%esp)
   0x08048f0a <+74>:    mov    0x1c(%esp),%eax
   0x08048f0e <+78>:    mov    %eax,0x8(%esp)
   0x08048f12 <+82>:    mov    0x1c(%esp),%eax
   0x08048f16 <+86>:    mov    %eax,0x4(%esp)
---Type <return> to continue, or q <return> to quit---
   0x08048f1a <+90>:    mov    0x1c(%esp),%eax
   0x08048f1e <+94>:    mov    %eax,(%esp)
   0x08048f21 <+97>:    call   0x8054700 <setresgid>
   0x08048f26 <+102>:   mov    0x18(%esp),%eax
   0x08048f2a <+106>:   mov    %eax,0x8(%esp)
   0x08048f2e <+110>:   mov    0x18(%esp),%eax
   0x08048f32 <+114>:   mov    %eax,0x4(%esp)
   0x08048f36 <+118>:   mov    0x18(%esp),%eax
   0x08048f3a <+122>:   mov    %eax,(%esp)
   0x08048f3d <+125>:   call   0x8054690 <setresuid>
   0x08048f42 <+130>:   lea    0x10(%esp),%eax
   0x08048f46 <+134>:   mov    %eax,0x4(%esp)
   0x08048f4a <+138>:   movl   $0x80c5348,(%esp)
   0x08048f51 <+145>:   call   0x8054640 <execv>
   0x08048f56 <+150>:   jmp    0x8048f80 <main+192>
   0x08048f58 <+152>:   mov    0x80ee170,%eax
   0x08048f5d <+157>:   mov    %eax,%edx
   0x08048f5f <+159>:   mov    $0x80c5350,%eax
   0x08048f64 <+164>:   mov    %edx,0xc(%esp)
   0x08048f68 <+168>:   movl   $0x5,0x8(%esp)
   0x08048f70 <+176>:   movl   $0x1,0x4(%esp)
   0x08048f78 <+184>:   mov    %eax,(%esp)
   0x08048f7b <+187>:   call   0x804a230 <fwrite>
   0x08048f80 <+192>:   mov    $0x0,%eax
---Type <return> to continue, or q <return> to quit---
```

```
    0x08048f85 <+197>:   leave
    0x08048f86 <+198>:   ret
```

여기서 `0x08048ed4 <+20>: call   0x8049710 <atoi>`에서 `atoi`를 통해 나온 결과 값이 `0x08048ed9 <+25>: cmp $0x1a7,%eax` 부분에서 비교한다는 점을 알게되었고 `'1a7'` 값이 10진수에서 `'423'` 라는 것을 통해서 이 값을 인자로 넣어봤다.

그렇게 되면 입력을 받게 되는 다른 상태로 넘어가는 줄 알았지만 `whoami` 명령어로 어떤 사용자인지 알아보면 내가 level1이 되어있음을 알 수 있었다.

```
level0@RainFall:~$ ./level0  423
$ whoami
level1
```

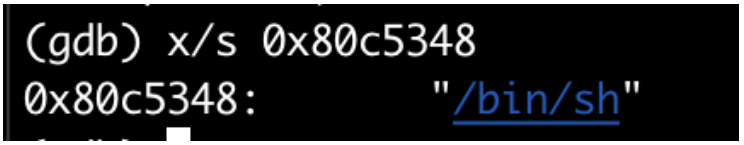
이를 통해서 `/home/user/level1`에 갈 수 있는 권한이 생겼다는 것을 알게 되었고 이 곳에서 ls -al를 통해서 숨겨진 파일을 찾은 후 `.pass` 라는 파일을 읽었을 때 정답을 알 수 있었다

▼ *추가설명 : 아래는 execv를 호출하기 전 내부적으로 동작하는 어셈블리어 코드이다*

```
0x08048ee0 <+32>:   movl   $0x80c5348,(%esp)
//0x80c5348 주소값에 저장된 문자열을 esp에 저장
0x08048ee7 <+39>:   call   0x8050bf0 <strdup>
//문자열을 복제해서 저장
0x08048eec <+44>:   mov    %eax,0x10(%esp)
//반환된 문자열을 esp+16에 저장
0x08048ef0 <+48>:   movl   $0x0,0x14(%esp)
// esp+20에 NULL저장


0x08048f42 <+130>:  lea    0x10(%esp),%eax
//레지스터에 스택포인터+16 값을 저장함
0x08048f46 <+134>:  mov    %eax,0x4(%esp)
//레지스터 값을 스택포인터+4에 저장함
0x08048f4a <+138>:  movl   $0x80c5348,(%esp)
//0x80c5348 주소값에 저장된 문자열을 esp에 저장
//아래 첨부한 이미지를 보면 "/bin/sh"을 저장하고 있다
0x08048f51 <+145>:  call   0x8054640 <execv>
//
//
[esp]       - 0x80c5348 (pointer to "/bin/sh")
[esp + 4]   - esp + 16 (pointer to [argv[0], NULL])
[esp + 8]   - not used
[esp + 12]  - not used
[esp + 16]  - 0x80c5348 (pointer to "/bin/sh")
[esp + 20]  - 0x0 (NULL)
```

```
//
결론적으로 execv("/bin/bash", ["/bin/sh", "NULL"])이 실행된다
```




```
$ cd /home/user/level1
$ ls -al
total 17
dr-xr-x---+ 1 level1 level1   80 Mar  6  2016 .
dr-x--x--x  1 root   root    340 Sep 23  2015 ..
-rw-r--r--  1 level1 level1  220 Apr  3  2012 .bash_logout
-rw-r--r--  1 level1 level1 3530 Sep 23  2015 .bashrc
-rwsr-s---+ 1 level2 users  5138 Mar  6  2016 level1
-rw-r--r--+ 1 level1 level1   65 Sep 23  2015 .pass
-rw-r--r--  1 level1 level1  675 Apr  3  2012 .profile
$ cat .pass
1fe8a524fa4bec01ca4ea2a869af2a02260d4a7d5fe7e7c24d8617e6dca12d3a
```

정답

`1fe8a524fa4bec01ca4ea2a869af2a02260d4a7d5fe7e7c24d8617e6dca12d3a`

출처

[LINUX] 세그멘테이션 오류(Core Dumped)

분명 환경에서 잘 되던 것 같은 문제가... 갑자기 이런 에러가 뜨며 죽는 경우를 봤다. 해결하기 전, 검색...

https://m.blog.naver.com/rtyuipl/222507009548