

DEFCON 2009

Binary L33tness 100, EDB로 풀어보자.

이름: 김 연재 (<http://hisjournal.net/blog>)

소속: CERT-IS (<http://www.cert-is.com>)

Table of Contents

- 준비 사항
- 바이너리 속을 볼까?
- 다시 한 번 정리
- 참고 문헌

```
yeonJae@hishome: ~/Workspace/BL100
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
yeonjae@hishome:~/Workspace/BL100$ ./defcon17_BL_100
What is the password? 1234
I smack you down. Step off bitch.
yeonjae@hishome:~/Workspace/BL100$
```

지난 6~8일에 있었던 데프콘 CTF 예선 문제입니다. 위 그림처럼 실행하면 암호를 묻고 틀린다면, 욕을 하는 기분 나쁜 녀석입니다. 기분도 나빠졌으니 까짓꺼 한 번 깨줘야겠죠?

준비 사항

- 리눅스 계열의 OS.
- libnet1 패키지
- Evans Debugger

이 파일은 ELF 포맷으로 되어 있습니다. ELF란 유닉스, 리눅스 등의 실행 파일의 구조인데, 이에 대해 설명하려면 끝이 없으니 자세한 내용을 알고 싶으시면 아래 참고 문헌을 읽어주세요. 아무튼 이 녀석에 대해 파해칠려면 리눅스가 필요하지요. 그리고 이 녀석을 실행하려면 libnet1 패키지가 설치되어야 합니다.

```
sudo apt-get install libnet1
```

전 이 문제를 디버깅툴로 풀려고 합니다. 리눅스에는 EDB라는 윈도우즈의 OllyDbg와 유사한 인터페이스를 갖춘 GUI 디버깅툴이 있습니다. 실제로 EDB 개발자는 OllyDbg를 모델로 하여 개발하였다고 하더군요. 아래 참고 문헌을 참조하여 설치해주세요.

바이너리 파일 속을 볼까?

준비 다 되었으면 이제 본격적으로 시작해볼까요?

리눅스에서는 strings 도구를 사용하여 파일에 포함되어 있는 문자열들을 뽑아낼 수 있습니다. 이것은 실행 파일도 마찬가지지요. 뽑아낸 문자열에는 라이브러리 이름, 함수 이름, 실제로 이용되는 문자열 등 등 아주 많은 정보가 포함됩니다. 더 자세한 내용은 man 페이지에서... ^^;

```
strings defcon17_BL_100
```



```
yeonjae@hishome: ~/Workspace/BL100
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
H+|$,
$Info: This file is packed with the UPX executable packer http://upx.sf.net $
$Id: UPX 3.01 Copyright (C) 1996-2007 the UPX Team. All Rights Reserved. $
PROT_EXEC|PROT_WRITE failed.
/proc/se
lf/exe
[jUX
P05<
og      W
)3[^8U
UPX!u
lum9
"]8a
X"Jx
PB{k
@bQs
A@]P
]FWs
B      (h
 fi-0
:jLl
\ox
UPX!
yeonjae@hishome:~/Workspace/BL100$
```

그런데 문자열을 뽑아내었는데도 원하는 정보가 나오지 않습니다. 위 그림처럼 단편적으로 쪼개진 의미 없는 문자열만 나오지요. 다만, 이해할 수 있는 문장이 하나 나오는데, 내용인 즉슨 "이 파일은 UPX로 압축되었다"라고 나옵니다. 압축을 풀어야 이 파일을 뜯어볼 수 있다는 얘기지요. 다행히 UPX는 -d 옵션을 이용하여 쉽게 풀 수 있습니다.

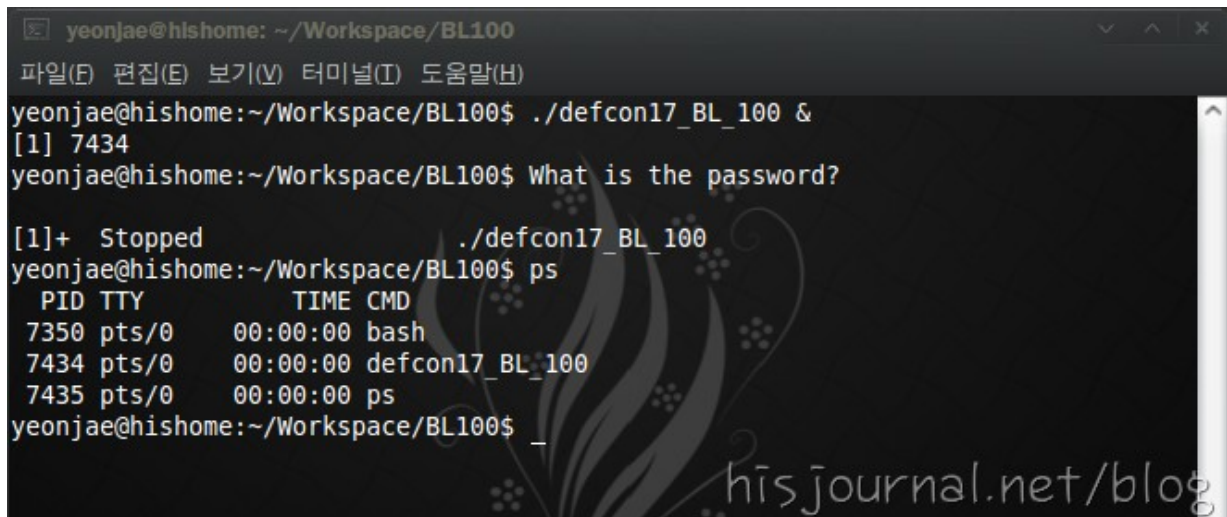
```
upx -d defcon17_BL_100
```

그러나 계속 찾아오는 시련... 황당하게도 체크섬 오류로 압축을 풀 수 없다고 나옵니다. 여기서 GG 처야하는건지... 압축을 못 풀면 내부 정보를 알 수 없는데 어떻게 풀라는 말인지?

하지만 방법이 있더군요. UPX든 뭐든 간에 아무리 실행 파일을 압축했어도 실행되는 당시에는 압축이 풀려 있어야 실행이 되겠지요. 아니면 컴퓨터가 어떻게 알아 먹고 실행하겠어요? 그리고 아마 메모리 어딘가에 압축이 풀린 프로세서가 떠 있을 겁니다. 우리는 그 녀석을 낚아서 회를 치는 겁니다.

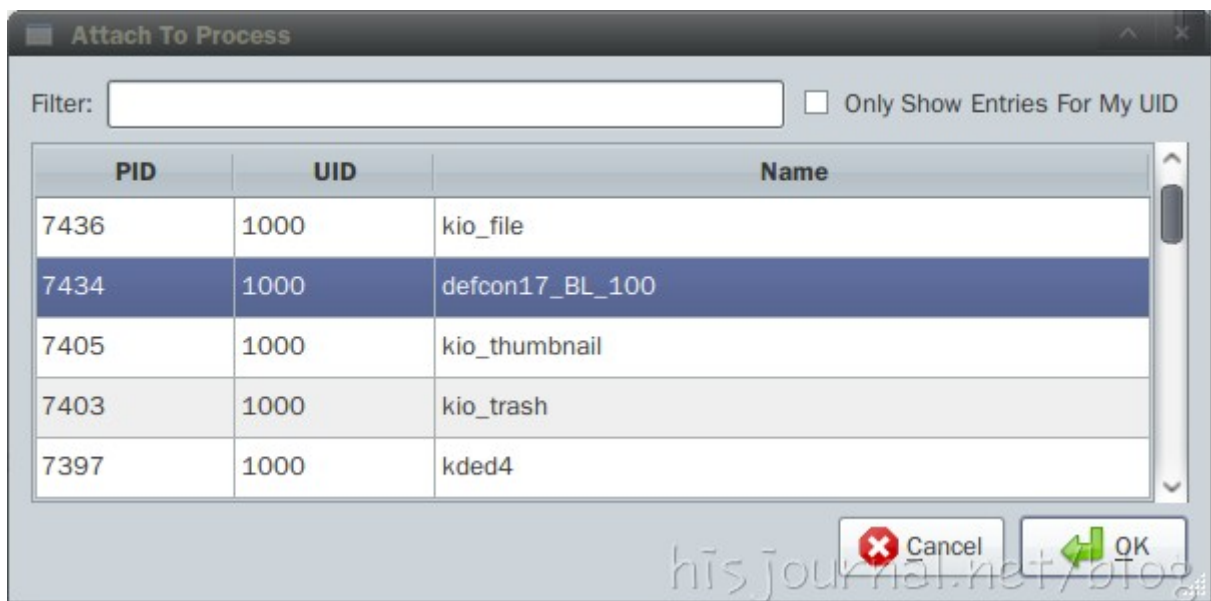
그럼 어떻게 낚을까요? 낚시에서는 미끼를 던지는 게 기본이겠죠? 미끼를 한 번 쥐 볼까요?

```
./defcon17_BL_100 &
```



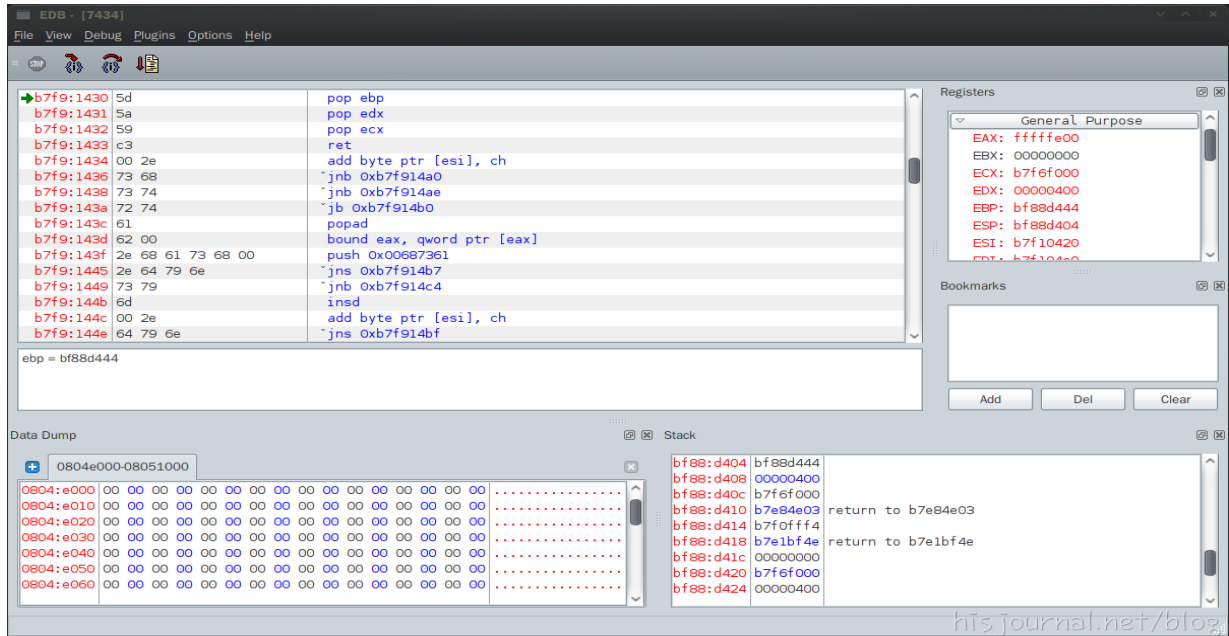
```
yeonjae@hishome: ~/Workspace/BL100
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
yeonjae@hishome:~/Workspace/BL100$ ./defcon17_BL_100 &
[1] 7434
yeonjae@hishome:~/Workspace/BL100$ What is the password?
[1]+  Stopped                  ./defcon17_BL_100
yeonjae@hishome:~/Workspace/BL100$ ps
  PID TTY          TIME CMD
 7350 pts/0    00:00:00 bash
 7434 pts/0    00:00:00 defcon17_BL_100
 7435 pts/0    00:00:00 ps
yeonjae@hishome:~/Workspace/BL100$ _
```

파일을 실행할 때 뒤에 &(앰퍼센트)를 붙이면 실행 된 이후에 프로세서를 백그라운드로 넘겨서 대기시킵니다. ps 명령어로 백그라운드에 대기중인 녀석을 확인할 수 있지요. 그런 다음에 EDB를 실행하여 프로세서를 불러옵니다.

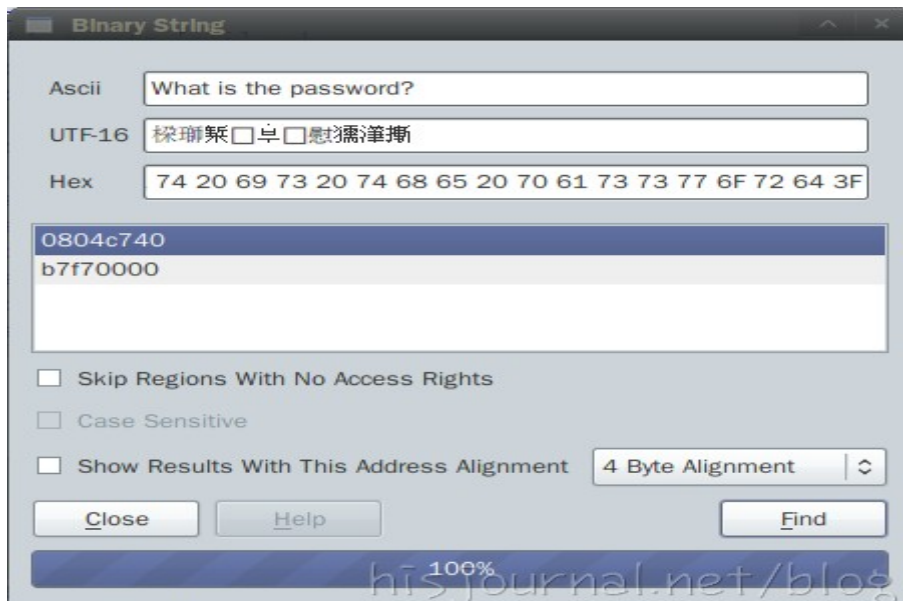


하지만 프로세서가 불러오지 않을 겁니다. 녀석이 아직 미끼를 안 물었거든요. 미끼를 물게 하기 위해서는 bg 명령어를 씁니다. 백그라운드에 있는 작업을 재실행하라는 명령어지요.

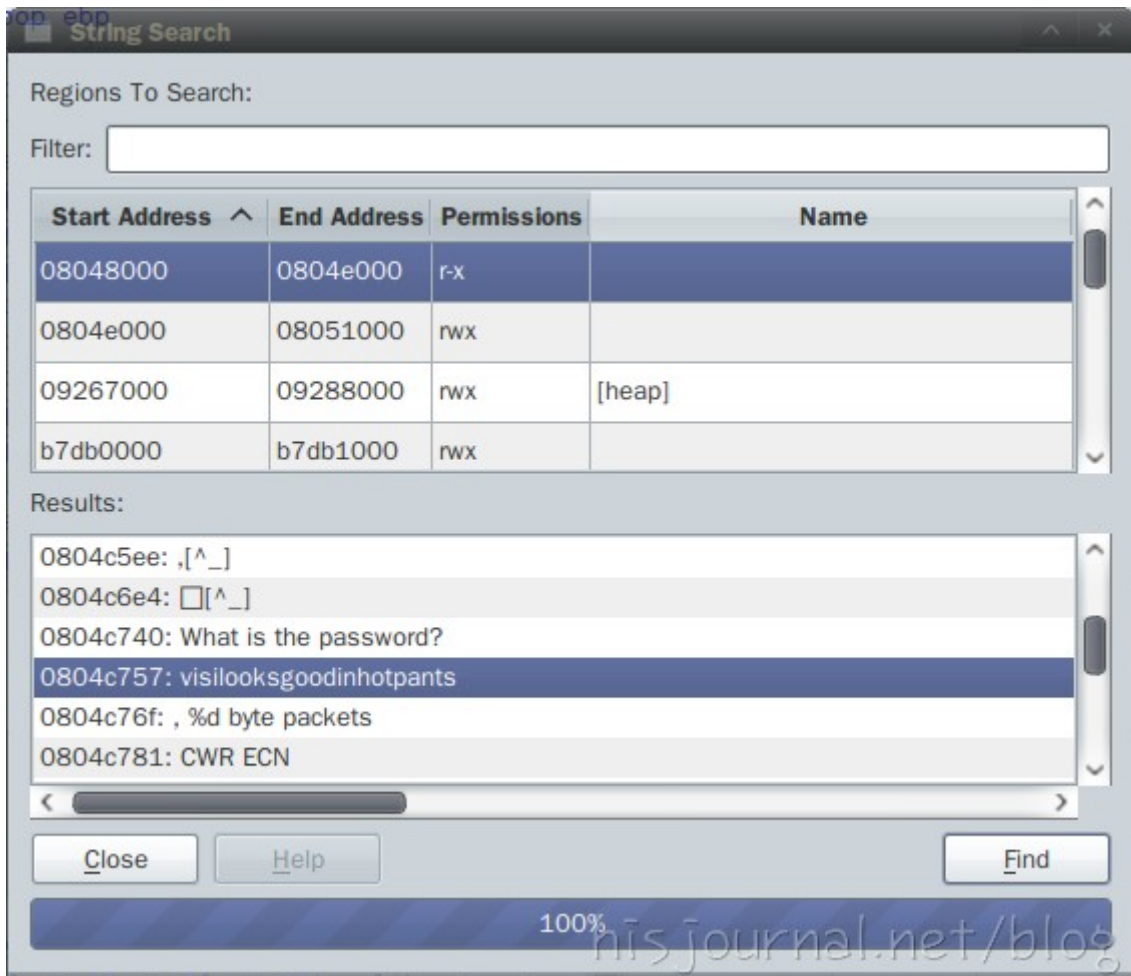
bg



그러면 녀석은 미끼를 덩씩 물어서 위 그림처럼 낚이는 거죠. 우리는 이것을 이용해 회를 치면 되는 겁니다. 우선, EDB에서 "메뉴 -> Plugins -> BinarySearcher -> Binary String Search"를 이용해 처음 시작했을 때 암호를 묻는 문자열을 검색합니다. 아마 그 문자열 근처에 비교할 암호가 있다거나 알고리즘이 있겠지요.



주소를 찾았으면 직접 찾아가 봅니다. 위에서 나온 주소를 더블클릭 해도 되고, "메뉴 -> Plugins -> StringSearcher -> String Search"를 이용해도 됩니다.



여기까지 문제없이 따라오셨다면 위 그림처럼 웬지 암호로 보이는 문자열이 나타납니다. 실제로 확인해 보죠.



빙고! 녀석을 완벽하게 회를 쳤습니다. 녀석이 겁을 먹었는지 저더러 "니가 내 애비다"라고 말하는군요. 훗! 처음부터 욕을 하지 말던가...

다시 한 번 정리

지금까지 내용을 간략히 정리하면 다음과 같습니다.

- 어떤 파일(프로그램)인지 알아보기 위한 기초조사 (strings, file, readelf, ...)
- 패킹 여부 확인 & 언패킹 (upx, ...)
- 프로세서를 백그라운드에 대기 (&, ps, bg, ...)
- EDB 등의 툴로 디버깅

정리해보니 너무나 간단하네요. 다만, 어떤 도구를 언제 어떻게 써야 하는지만 알면 되는거죠. 기초적인 ELF가 뭔지, UPX가 뭔지 등은 검색해보면 다 나오구요.

제가 한 방법과 다르게 푸는 방법도 있습니다. 그 방법은 아래 참고 문헌에 링크를 걸어놓겠습니다. 중국의 해커가 자신의 블로그에 올렸네요. 그 방법은 디버깅툴을 쓰지 않고 리눅스의 도구들을 이용합니다. 프로세서를 낚시하는 것은 똑같은데 낚는 방법이 다르지요.

참고 문헌

ELF File Format :: <http://kldp.org/node/103483>

EDB, Evan's Debugger :: <http://linux-virus.springnote.com/pages/3512025>

UPX :: <http://en.wikipedia.org/wiki/UpX>

Defcon 17 Wargame 競賽解題心得 ::

<http://timhsu.chroot.org/2009/06/defcon17wargame.html>