

Outrepasser l'authentification par reverse engineering - Linux x86 -

Jonathan Salwan #
submit AT shell-storm.org #
<http://www.shell-storm.org> #

09/05/2009 #

Dans ce tutoriel qui est la suite de « **Outrepasser l'authentification par buffer overflow** » nous allons vous montrer comment bypasser cette authentification même s'il est impossible d'appliquer un buffer overflow.

Pour commencer il va nous falloir un petit outil:

- Hte** (Il nous permettra de rééditer le programme)
- Gdb** (débugger)

Pour l'installer sous des dérivés de Debian:

```
root@laptop:/home/# apt-get install ht
```

Prenons comme pour l'article précédent le même code source.

```
#include <stdio.h>
#include <stdlib.h>

void gestion()
{
    fprintf(stdout,"Bravo vous êtes dans la partie autorisée\n");
    exit(0);
}

int main(void)
{
    char pwd[10];
    printf("Password: ");
    scanf("%s", &pwd);
    if(!strcmp(pwd, "toto"))
        { gestion(); }
    else
        fprintf(stderr,"Password invalide!\n");

    return 0;
}
```

Une fois votre code compilé nous allons l'ouvrir avec **gdb** puis faire un **disass** sur le **main**.

(gdb) disass main

Dump of assembler code for function main:

[...]

```
0x080485a6 <main+43>:    mov    %eax,0x4(%esp)
0x080485aa <main+47>:    movl   $0x8048716,(%esp)
0x080485b1 <main+54>:    call   0x8048428 <scanf>    <==Appel fonction scanf
0x080485b6 <main+59>:    movl   $0x8048719,0x4(%esp)
0x080485be <main+67>:    lea    -0x12(%ebp),%eax
0x080485c1 <main+70>:    mov    %eax,(%esp)
0x080485c4 <main+73>:    call   0x8048468 <strcmp>  <==Appel fonction strcmp
0x080485c9 <main+78>:    test   %eax,%eax      <==Test stdin avec le password
0x080485cb <main+80>:    jne    0x80485d4 <main+89> <==Si password faux saut à @0x80485d4
0x080485cd <main+82>:    call   0x8048544 <gestion> <==Sinon saut à @gestion
0x080485d2 <main+87>:    jmp    0x80485f9 <main+126>
0x080485d4 <main+89>:    mov    0x804a040,%eax
[...]
```

Ce qui nous intéresse ici c'est de modifier l'instruction **jne**. Pour ça nous avons plusieurs solutions:

-Soit on remplace **jne** par **je**

-Soit on remplace **jne** par un **nop**

Informations instructions

JNE ZF = 0 jump if not equal

JE ZF = 1 jump if equal

NOP = No operation

Vous l'avez sûrement compris si on remplace donc **jne** par **je**.

A la saisie du password si celui-ci est faux notre programme va donc passer à l'instruction du dessous soit **call 0x8048544 <gestion>**.

Si nous remplaçons **jne** par un **nop**.

L'instruction **jne 0x80485d4** n'aura donc pas lieu et le programme exécutera directement **call 0x8048544 <gestion>**.

Avec un **nop** le programme ressemblerait à ça:

```
0x080485c4 <main+73>:    call   0x8048468 <strcmp>  <==Appel fonction strcmp
0x080485c9 <main+78>:    test   %eax,%eax      <==Test stdin avec le password
0x080485cb <main+80>:    nop                 <==pas opération
0x080485cd <main+82>:    call   0x8048544 <gestion> <==Saut à l'adresse gestion
```

Donc pas de test si le password est bon ou pas ;)

Mettons toutes nos théories en pratique, pour ça nous allons utiliser l'outil **h****t****e**.

Ouvrez donc votre programme avec **h****t****e** puis sélectionnez à l'aide de **F6** le mode (**elf/image**) et positionnez vous au niveau du main, ce qui vous donnera une chose comme ceci:

```
File Edit Windows Help Analyser 11:56 09.05.2009
[x] /home/submit/all/prog/tuto/renvers/main 2
<.text> @000005cb jnz 80485d4h
main+50
8048576 |   call      wrapper_804a01c_8048478
804857b |
..... | ; ****
..... | ; function main (global)
..... | ; ****
..... | main:                      ;xref o80484a7
..... |   lea       ecx, [esp+4]
804857f |   and      esp, offfffffoh
8048582 |   push     dword ptr [ecx-4]
8048585 |   push     ebp
8048586 |   mov      ebp, esp
8048588 |   push     ecx
8048589 |   sub      esp, 24h
804858c |   mov      eax, gs:[14h]
8048592 |   mov      [ebp-8], eax
8048595 |   xor      eax, eax
8048597 |   mov      dword ptr [esp], strz_Password:_804870b
804859e |   call    wrapper_804a00c_8048438
80485a3 |   lea       eax, [ebp-12h]
80485a6 |   mov      [esp+4], eax
80485aa |   mov      dword ptr [esp], data_8048716
80485b1 |   call    wrapper_804a008_8048428
80485b6 |   mov      dword ptr [esp+4], strz_toto_8048719
80485be |   lea       eax, [ebp-12h]
80485c1 |   mov      [esp], eax
80485c4 |   call    wrapper_804a018_8048468
80485c9 |   test    eax, eax
80485cb |   jnz     loc_80485d4
80485cd |   call    gestion
80485d2 |   jmp     loc_80485f9
80485cb/000005cb
```

1help 2save 3open 4edit 5goto 6mode 7search 8symbols 9viewin. 0quit

Avec la touche **F4** (**edit**) nous allons remplacer l'instruction **jne** (0x75) par **je** (0x74) puis sauvegardons avec **F2**.

Ensuite testons notre programme.

```
root@laptop:/home/submit/all/prog/tuto/renvers# ./main
Password: a
Bravo vous êtes dans la partie autorisée
root@laptop:/home/submit/all/prog/tuto/renvers#
```

Notre programme a bien appelé la fonction gestion même si le password était faux, testons avec le vrai password.

```
root@laptop:/home/submit/all/prog/tuto/renvers# ./main
Password: toto
Password invalide!
root@laptop:/home/submit/all/prog/tuto/renvers#
```

Maintenant testons notre théorie avec les **nop** (0x90) ce qui va donner une chose comme ceci:

The screenshot shows the assembly view of the 'main' function in the 'renvers' program. The assembly code is as follows:

```
File Edit Windows Help Analyser 12:46 09.05.2009
[.]----- /home/submit/all/prog/tuto/renvers/main -----2
<.text> @000005cb  nop
main+50
8048576 | e8fdffff    call      wrapper_804a01c_80484>
804857b |
..... | ; ****
..... | ; function main (global)
..... | ; ****
..... | main:           ;xref o80484a7
..... | 8d4c2404        lea       ecx, [esp+4]
804857f | 83e4f0        and       esp, offffffh
8048582 | ff71fc        push      dword ptr [ecx-4]
8048585 | 55             push      ebp
8048586 | 89e5            mov      ebp, esp
8048588 | 51             push      ecx
8048589 | 83ec24        sub      esp, 24h
804858c | 65a114000000    mov      eax, gs:[14h]
8048592 | 8945f8        mov      [ebp-8], eax
8048595 | 31c0            xor      eax, eax
8048597 | c704240b870408  mov      dword ptr [esp], strz>
804859e | e895feffff    call     wrapper_804a00c_80484>
80485a3 | 8d45ee        lea       eax, [ebp-12h]
80485a6 | 89442404    mov      [esp+4], eax
80485aa | c7042416870408  mov      dword ptr [esp], data>
80485b1 | e872feffff    call     wrapper_804a008_80484>
80485b6 | c744240419870408  mov      dword ptr [esp+4], st>
80485be | 8d45ee        lea       eax, [ebp-12h]
80485c1 | 890424        mov      [esp], eax
80485c4 | e89ffeffff    call     wrapper_804a018_80484>
80485c9 | 85c0            test    eax, eax
80485cb | 90             nop
80485cc | 90             nop
80485cd | e872ffffff    call     gestion
80485d2 | eb25            jmp     loc_80485f9
80485d4 |
```

The assembly code is mostly standard x86 assembly, with several NOP (opcode 90) instructions highlighted in red. These NOPs are placed at various points in the assembly, such as after the function prologue and before the call to the 'gestion' function. The Immunity Debugger interface shows the assembly code in the main window, with various registers and memory dump tabs visible at the bottom.

Sauvegardons avec la touche **F2** puis testons.

```
root@laptop:/home/submit/all/prog/tuto/renvers# ./main  
Password: a  
Bravo vous êtes dans la partie autorisée  
root@laptop:/home/submit/all/prog/tuto/renvers#
```

Comme dans le premier exemple testons notre programme avec le vrai password.

```
root@laptop:/home/submit/all/prog/tuto/renvers# ./main  
Password: toto  
Bravo vous êtes dans la partie autorisée  
root@laptop:/home/submit/all/prog/tuto/renvers#
```

Conclusion :

S'il est impossible d'appliquer un buffer overflow pour passer l'authentification du programme le reverse engineering est une méthode imparable pour arriver à vos fins, excepté si le programme à pour uid(0). Dans ce cas là, pour éditer celui-ci, il vous faudra donc les droits root...

Shell-Storm.org