JAXON HAWS

INTRO TO BINARY EXPLOITATION

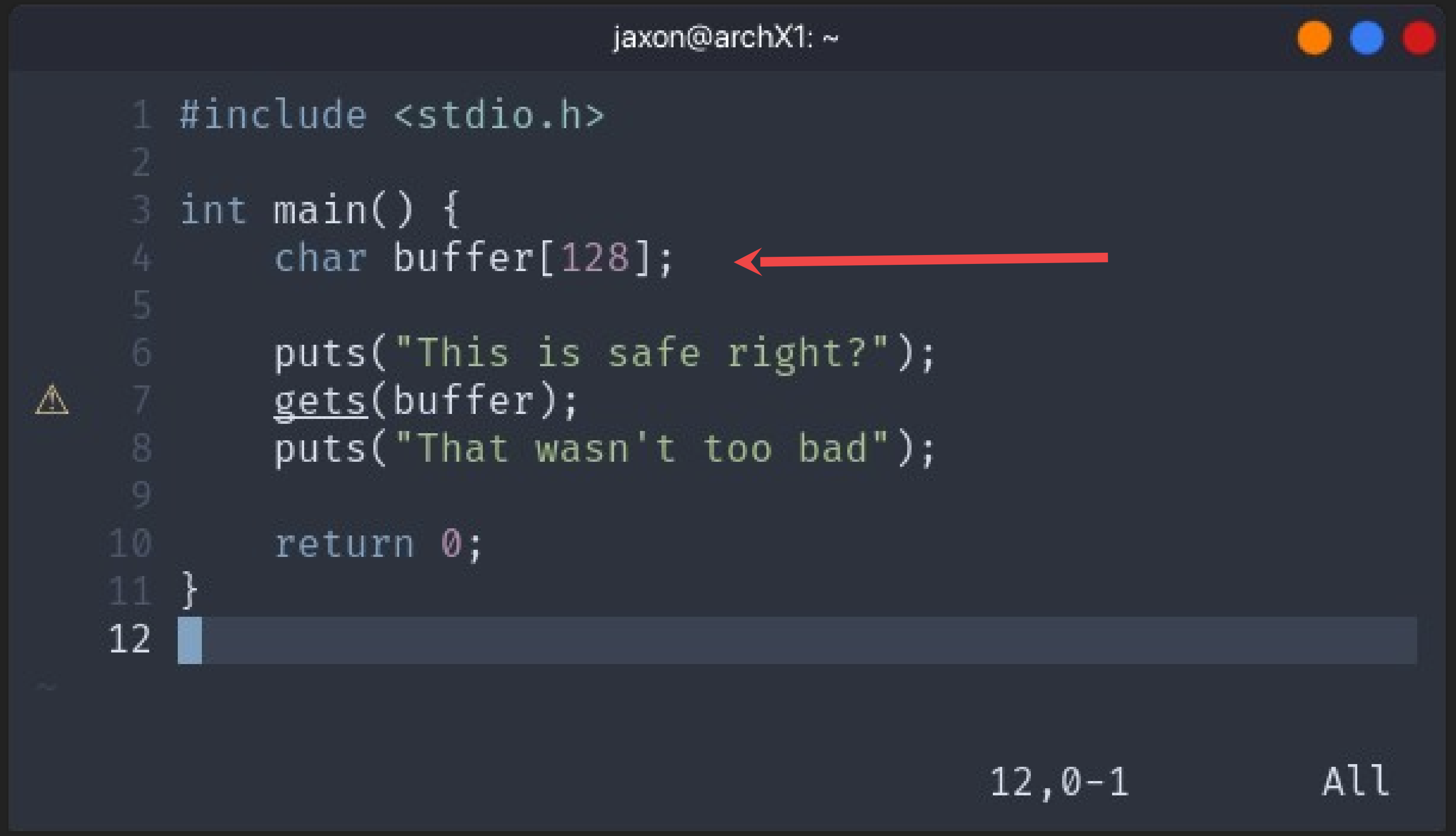
ABOUT ME

- 2nd Year Computer Science
- Clubs:
  - White Hat: Archivist -> President
  - CPLUG: Treasurer -> President
- Started racing motorcycles off-road this year

TABLE OF CONTENTS

WHAT IS A BUFFER?

- A buffer is basically an array
- Buffers must have a fixed size at compile time

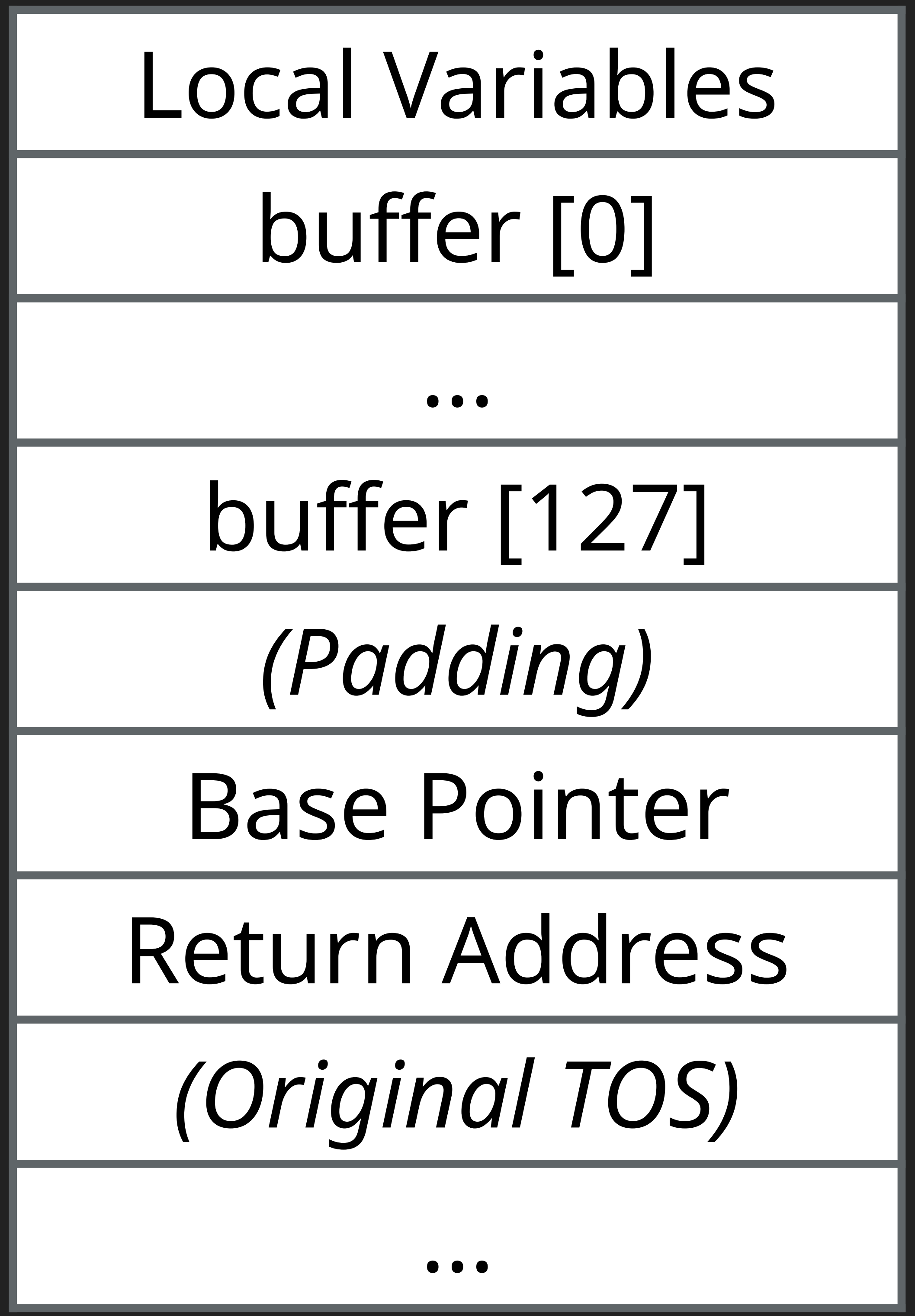


```
#include <stdio.h>

int main() {
    char buffer[128];

    puts("This is safe right?");
    gets(buffer);
    puts("That wasn't too bad");

    return 0;
}

```

WHERE DO BUFFERS LIVE?

- Buffers live on the stack
- C doesn't protect you
- Programmer's job to ensure that more than 128 characters don't get written to buffer

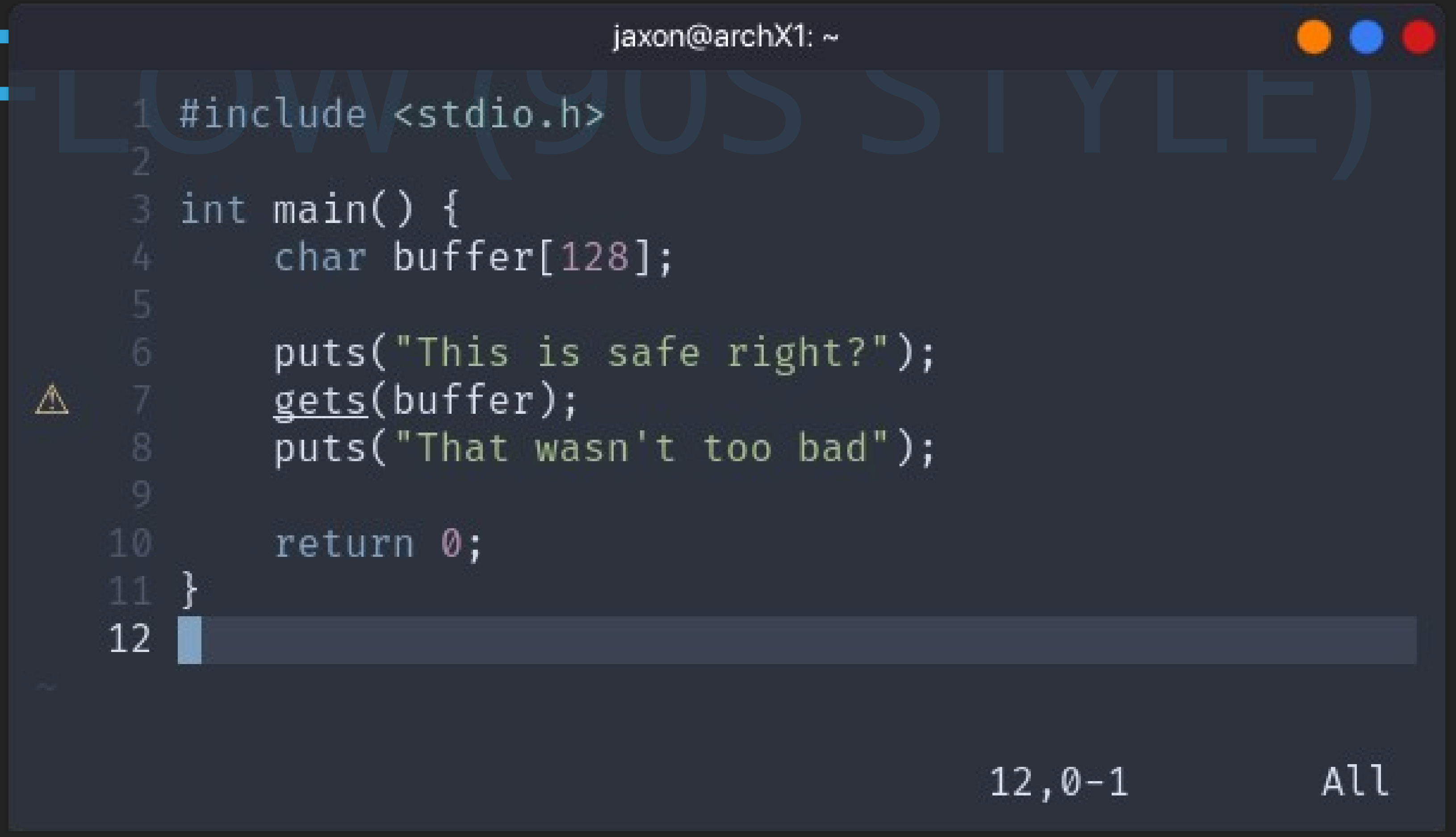
STACK FRAME



Grows upwards towards lower addresses

HOW TO EXPLOIT A BUFFER OVERFLOW (90S STYLE)

- Step 1: Find a buffer to overflow
- Step 2: Determine the buffer's size
- Step 3: Send data to overflow into the return address
- Step 4: Profit?

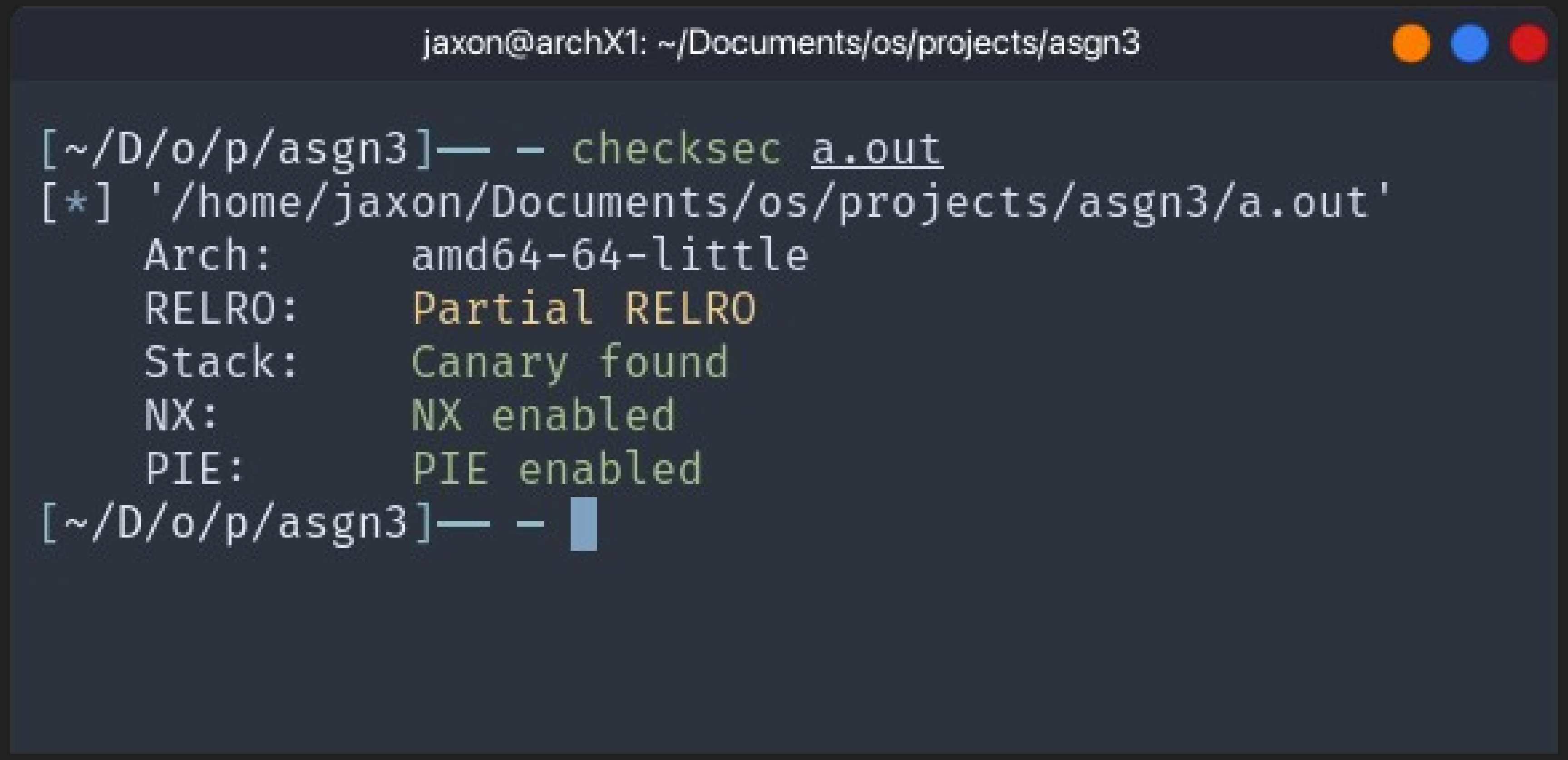


```
#include <stdio.h>

int main() {
    char buffer[128];

    puts("This is safe right?");
    gets(buffer);
    puts("That wasn't too bad");

    return 0;
}

```

```
12,0-1          All
```

MODERN SECURITY FEATURES

- Security has come a long way since the 90s
- Stack cookie/canary
- RELRO
- NX Bit
- PIE
- ASLR

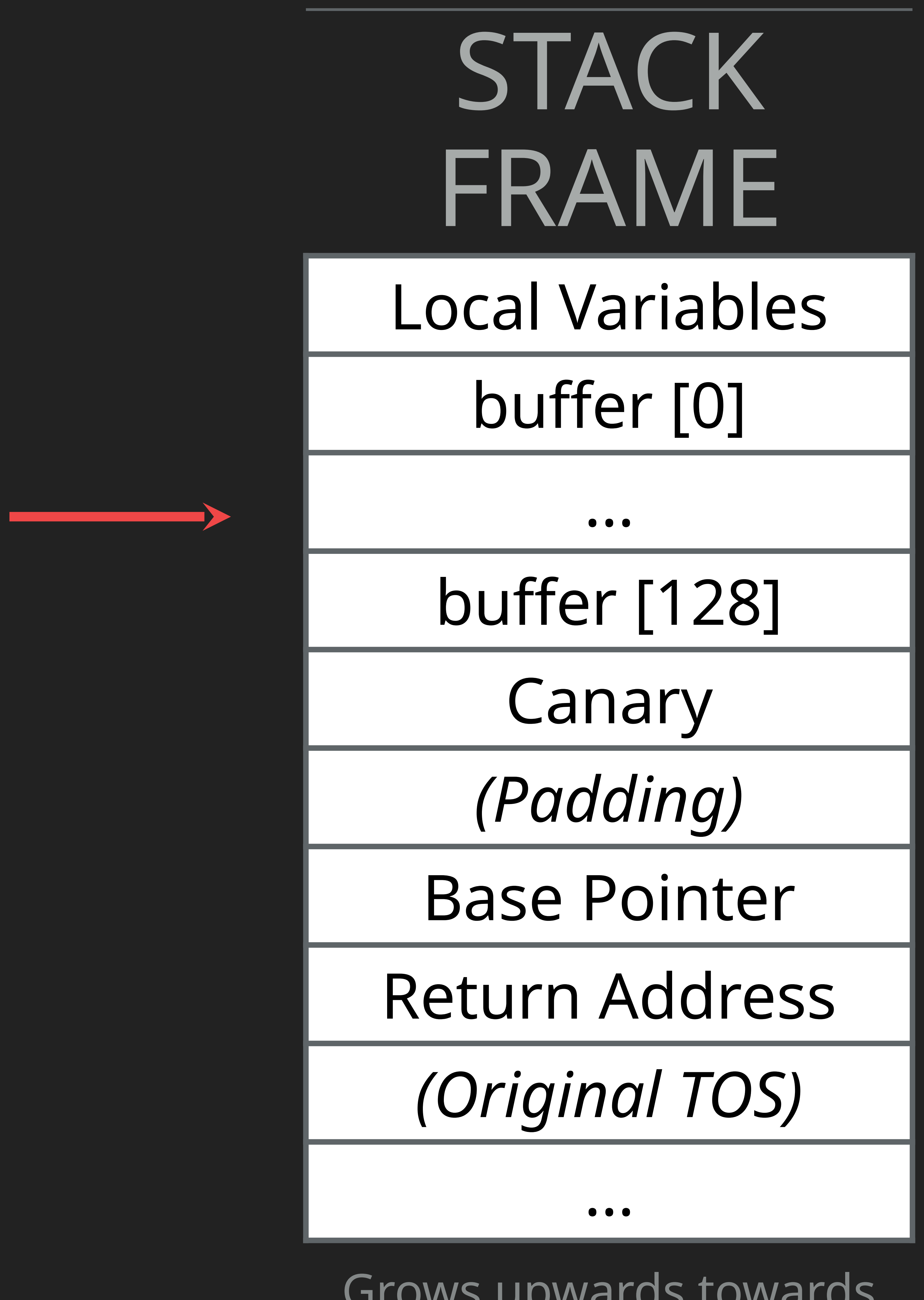


jaxon@archX1: ~/Documents/os/projects/asgn3

```
[~/D/o/p/asgn3]—— - checksec a.out
[*] '/home/jaxon/Documents/os/projects/asgn3/a.out'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[~/D/o/p/asgn3]—— -
```

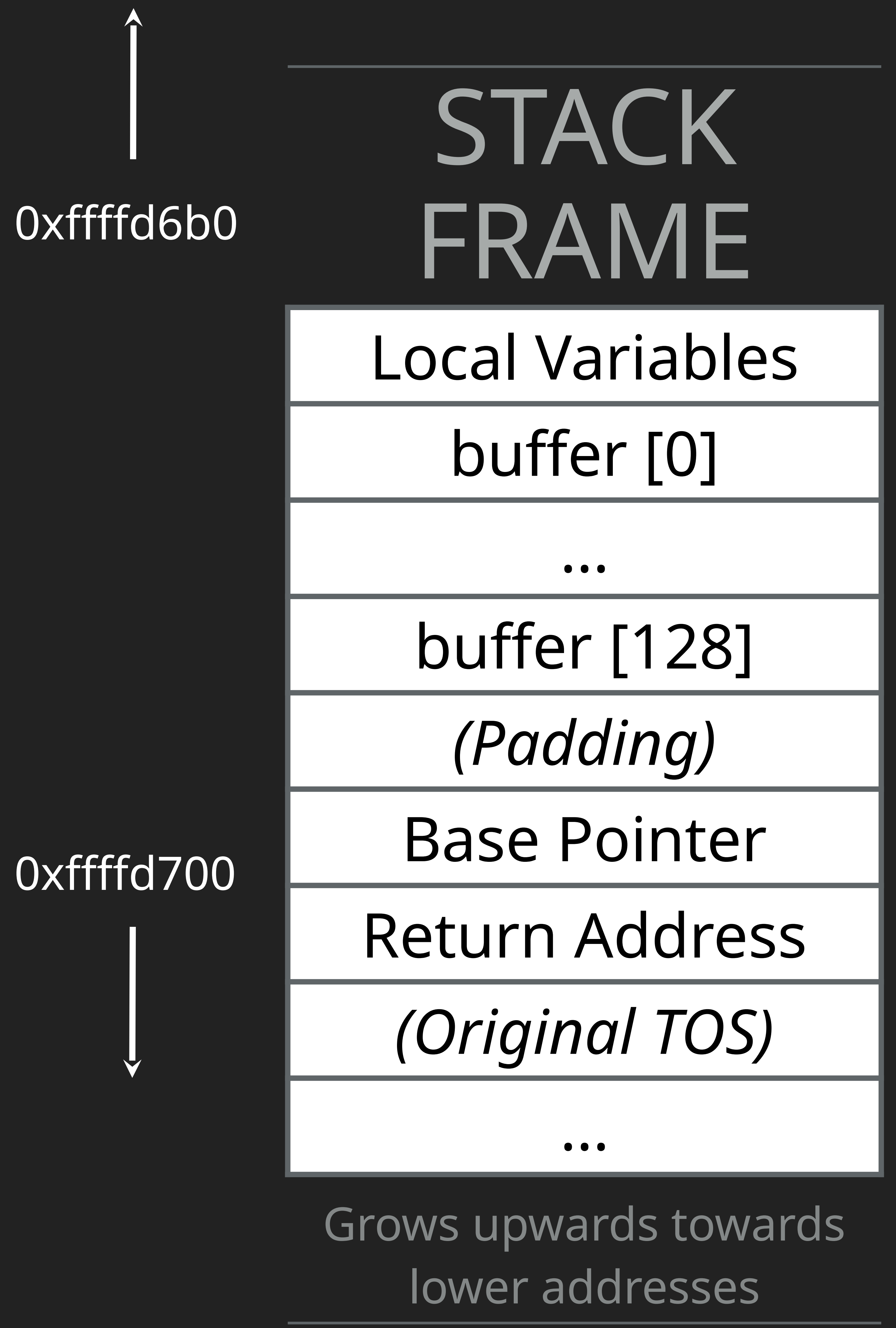
STACK CANARY/COOKIE

- Lives between locals and addresses
- Known value at start of runtime
- Value gets checked during runtime:
  - If values match: proceed
  - If values don't match: exit
- Enabled by default
- To disable: gcc overflow.c -fno-stack-protector
- Attack Style: Leak or brute force the cookie



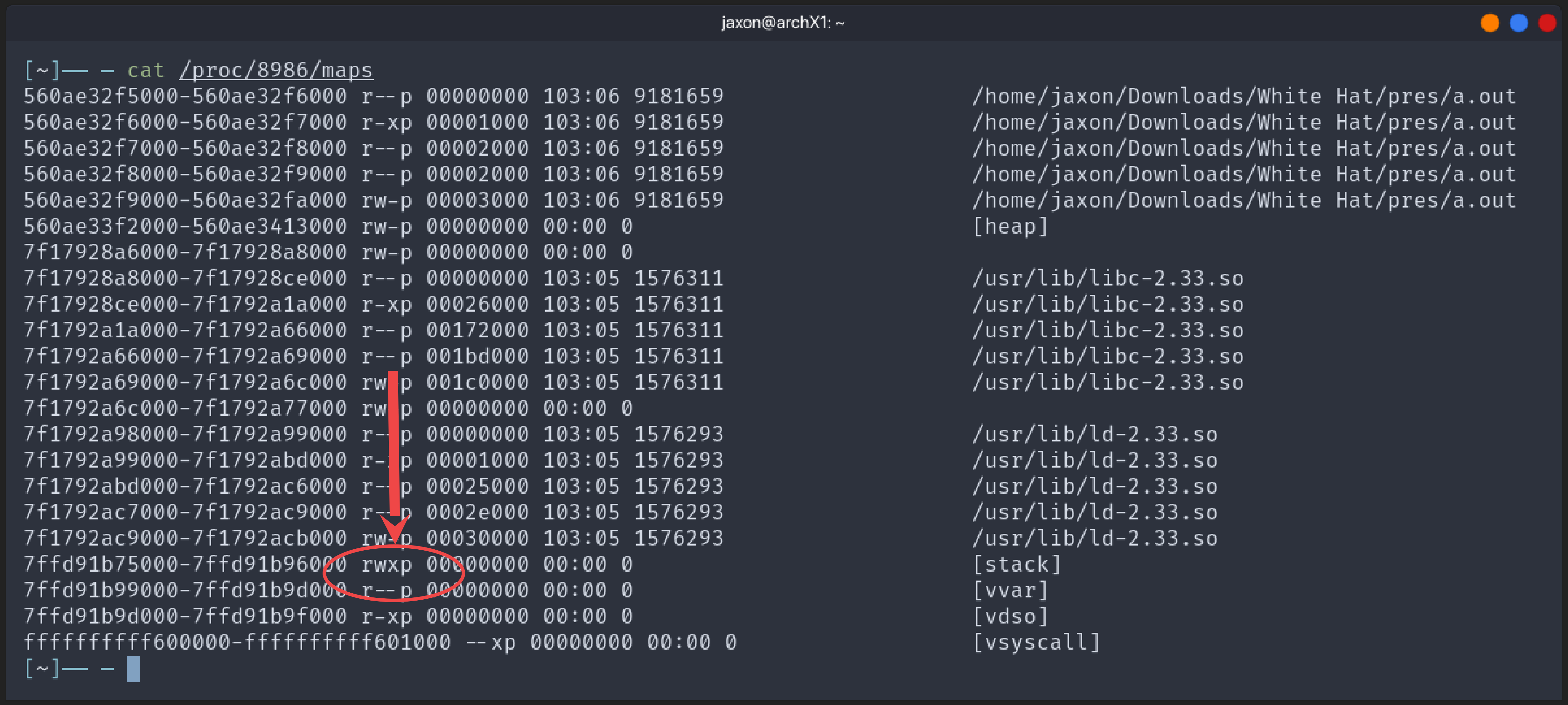
ASLR

- Address Space Layout Randomization
- Shifts stack addresses around at random
- Enabled by default on UNIX
- Not a compiler feature
- Attack Style: Don't hardcode addresses



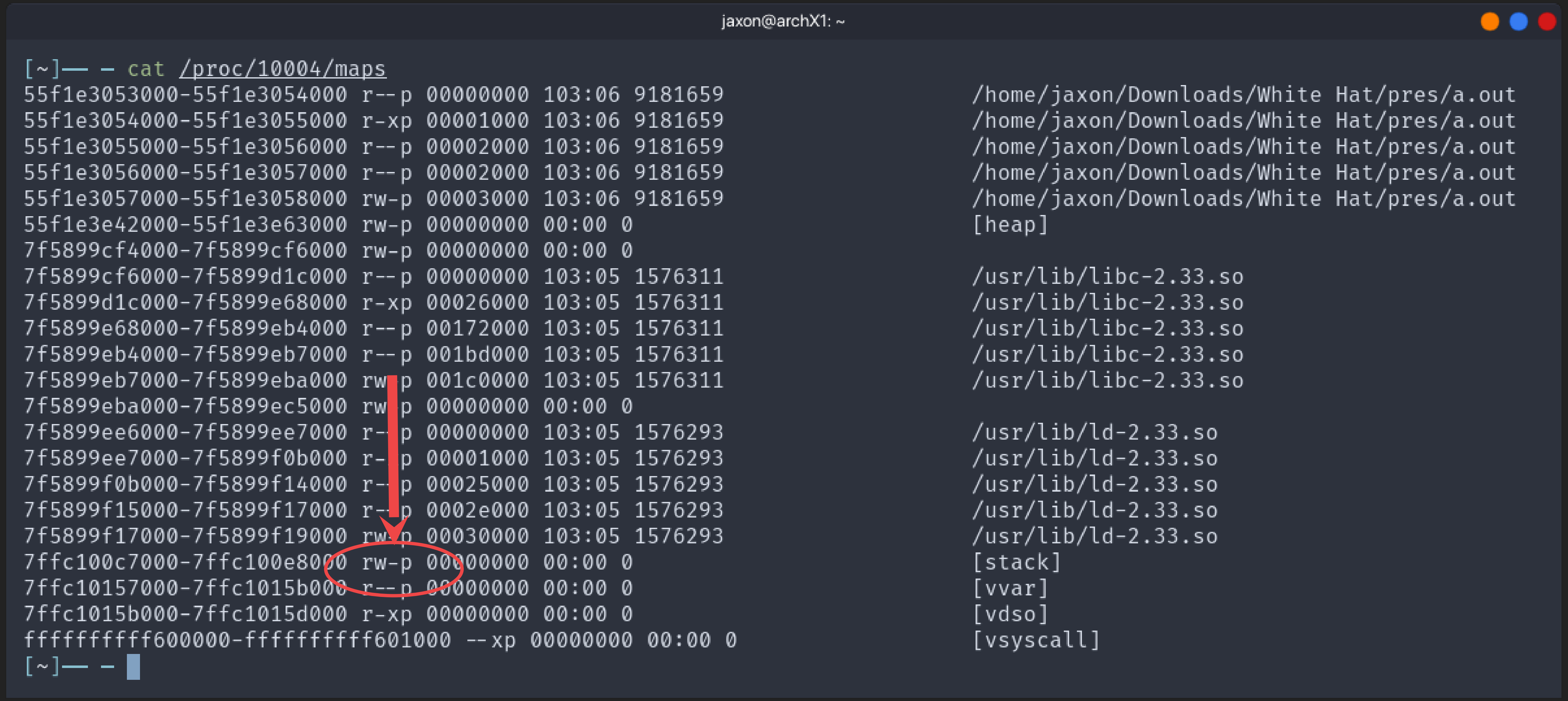
NX BIT

- No eXecute bit
- Memory is either:
  - read & execute
  - read & write
- Enabled by default
- To disable: gcc overflow.c -z execstack
- Attack Style: ROP



jaxon@archX1: ~

```
[~]── - cat /proc/8986/maps
560ae32f5000-560ae32f6000 r--p 00000000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
560ae32f6000-560ae32f7000 r-xp 00001000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
560ae32f7000-560ae32f8000 r--p 00002000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
560ae32f8000-560ae32f9000 r--p 00002000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
560ae32f9000-560ae32fa000 rw-p 00003000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
560ae33f2000-560ae3413000 rw-p 00000000 00:00 0                          [heap]
7f17928a6000-7f17928a8000 rw-p 00000000 00:00 0
7f17928a8000-7f17928ce000 r--p 00000000 103:05 1576311                  /usr/lib/libc-2.33.so
7f17928ce000-7f1792a1a000 r-xp 00026000 103:05 1576311                  /usr/lib/libc-2.33.so
7f1792a1a000-7f1792a66000 r--p 00172000 103:05 1576311                  /usr/lib/libc-2.33.so
7f1792a66000-7f1792a69000 r--p 001bd000 103:05 1576311                  /usr/lib/libc-2.33.so
7f1792a69000-7f1792a6c000 rw-p 001c0000 103:05 1576311                  /usr/lib/libc-2.33.so
7f1792a6c000-7f1792a77000 rw-p 00000000 00:00 0
7f1792a98000-7f1792a99000 r--p 00000000 103:05 1576293                  /usr/lib/ld-2.33.so
7f1792a99000-7f1792abd000 r-xp 00001000 103:05 1576293                  /usr/lib/ld-2.33.so
7f1792abd000-7f1792ac6000 r--p 00025000 103:05 1576293                  /usr/lib/ld-2.33.so
7f1792ac7000-7f1792ac9000 r--p 0002e000 103:05 1576293                  /usr/lib/ld-2.33.so
7f1792ac9000-7f1792acb000 rw-p 00030000 103:05 1576293                  /usr/lib/ld-2.33.so
7ffd91b75000-7ffd91b96000 rwxp 00000000 00:00 0                          [stack]
7ffd91b99000-7ffd91b9d000 r--p 00000000 00:00 0                          [vvar]
7ffd91b9d000-7ffd91b9f000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
[~]── -
```

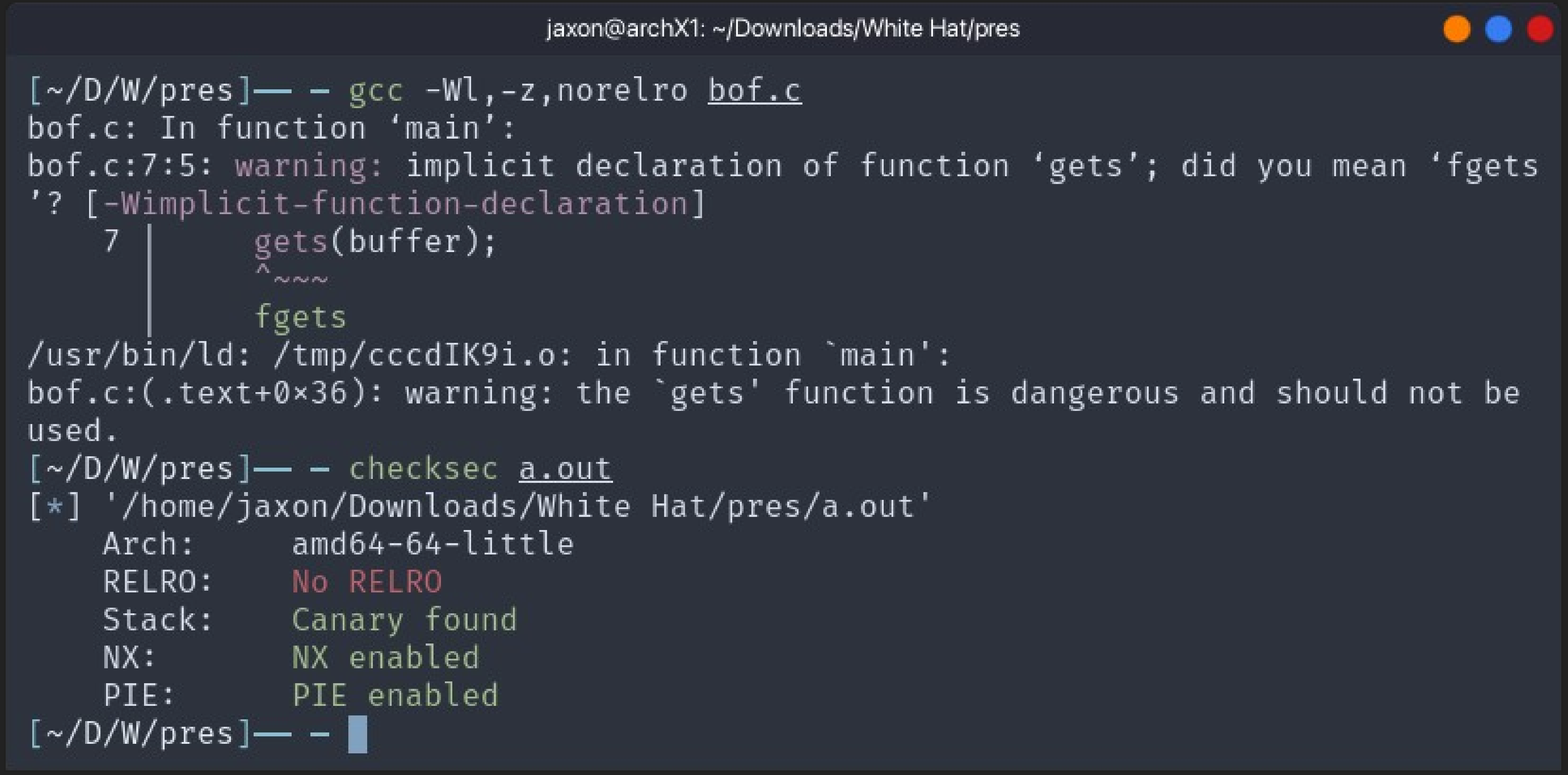


jaxon@archX1: ~

```
[~]── - cat /proc/10004/maps
55f1e3053000-55f1e3054000 r--p 00000000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
55f1e3054000-55f1e3055000 r-xp 00001000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
55f1e3055000-55f1e3056000 r--p 00002000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
55f1e3056000-55f1e3057000 r--p 00002000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
55f1e3057000-55f1e3058000 rw-p 00003000 103:06 9181659                  /home/jaxon/Downloads/White Hat/pres/a.out
55f1e3e42000-55f1e3e63000 rw-p 00000000 00:00 0                          [heap]
7f5899cf4000-7f5899cf6000 rw-p 00000000 00:00 0
7f5899cf6000-7f5899d1c000 r--p 00000000 103:05 1576311                  /usr/lib/libc-2.33.so
7f5899d1c000-7f5899e68000 r-xp 00026000 103:05 1576311                  /usr/lib/libc-2.33.so
7f5899e68000-7f5899eb4000 r--p 00172000 103:05 1576311                  /usr/lib/libc-2.33.so
7f5899eb4000-7f5899eb7000 r--p 001bd000 103:05 1576311                  /usr/lib/libc-2.33.so
7f5899eb7000-7f5899eba000 rw-p 001c0000 103:05 1576311                  /usr/lib/libc-2.33.so
7f5899eba000-7f5899ec5000 rw-p 00000000 00:00 0
7f5899ee6000-7f5899ee7000 r--p 00000000 103:05 1576293                  /usr/lib/ld-2.33.so
7f5899ee7000-7f5899f0b000 r-xp 00001000 103:05 1576293                  /usr/lib/ld-2.33.so
7f5899f0b000-7f5899f14000 r--p 00025000 103:05 1576293                  /usr/lib/ld-2.33.so
7f5899f15000-7f5899f17000 r--p 0002e000 103:05 1576293                  /usr/lib/ld-2.33.so
7f5899f17000-7f5899f19000 rw-p 00030000 103:05 1576293                  /usr/lib/ld-2.33.so
7ffc100c7000-7ffc100e8000 rw-p 00000000 00:00 0                          [stack]
7ffc10157000-7ffc1015b000 r--p 00000000 00:00 0                          [vvar]
7ffc1015b000-7ffc1015d000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
[~]── -
```

RELRO

- RELocation Read Only
- 2 Modes:
  - Partial (enabled by default, practically useless)
    - GOT comes before BSS segment
  - Full (not the default)
    - GOT becomes read only
- Attack Changes: Can't overflow global variables into GOT / Can't write to the GOT
- To disable: gcc -Wl,-z,norelro overflow.c



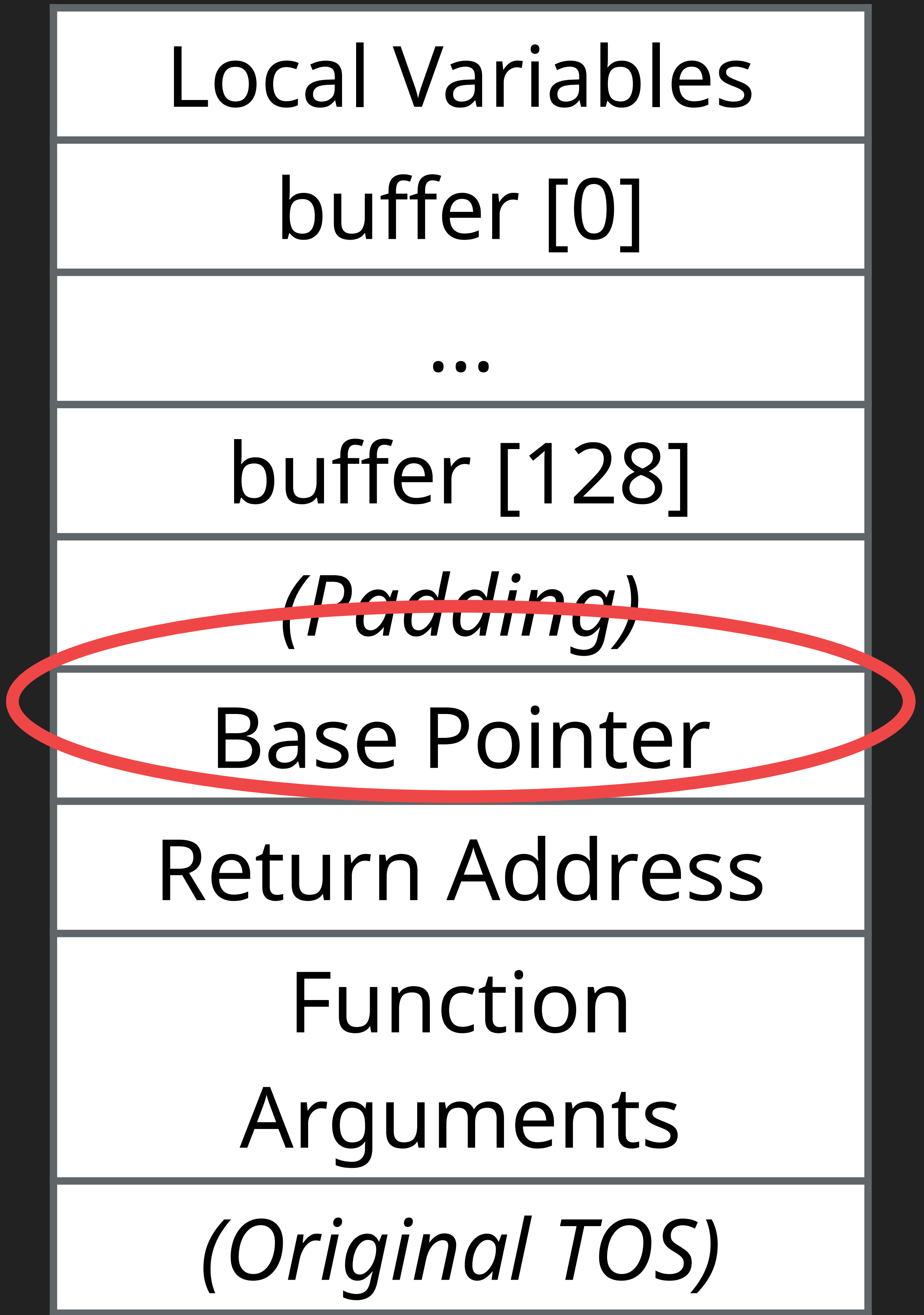
jaxon@archX1: ~/Downloads/White Hat/pres

```
[~/D/W/pres]── – gcc -Wl,-z,norelro bof.c
bof.c: In function ‘main’:
bof.c:7:5: warning: implicit declaration of function ‘gets’; did you mean ‘fgets
’? [-Wimplicit-function-declaration]
    7 |     gets(buffer);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/cccdIK9i.o: in function `main':
bof.c:(.text+0x36): warning: the `gets' function is dangerous and should not be
used.
[~/D/W/pres]── – checksec a.out
[*] '/home/jaxon/Downloads/White Hat/pres/a.out'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[~/D/W/pres]── –
```

32 BIT VS 64 BIT

- Register size
- 32 bit
  - Functions pass arguments on the stack
  - Fewer Registers than 64 bit
- 64 bit
  - Functions pass arguments through registers
  - Additional Registers
- Changes ROP approach

STACK FRAME



*SOME ASSEMBLY REQUIRED

DEMO TIME

REVIEW

- Check your buffers
- Use safe copying functions
- Use dynamic memory when size varies
- Modern problems require modern solutions

RESOURCES

- Canaries: http://phrack.org/issues/67/13.html
- Smashing the Stack for Fun and Profit: http://phrack.org/issues/49/14.html
- ROP: https://hovav.net/ucsd/dist/geometry.pdf
- Format String Attacks: https://seclists.org/bugtraq/2000/Sep/214
- Binary Security 101: https://ctf101.org/binary-exploitation/what-is-binary-security/
- CSC 429: Binary Exploitation

QUESTIONS?