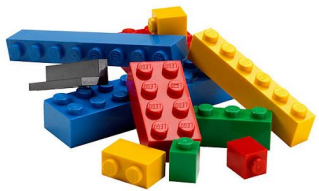
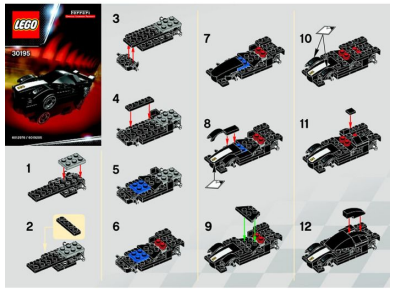
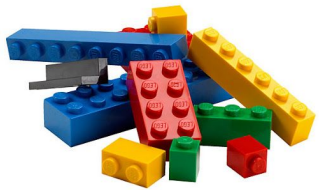
Module:
Return Oriented Programming

Binary Lego

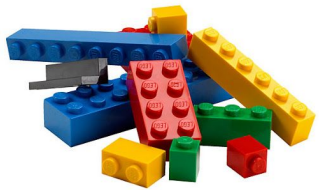
Yan Shoshitaishvili
Arizona State University




LEGO
30195
3
7
10
4
8
11
1
5
2
6
9
12




AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA

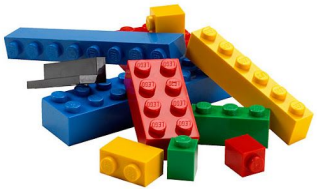


+


AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA

=

Bytes in
`.text`
section

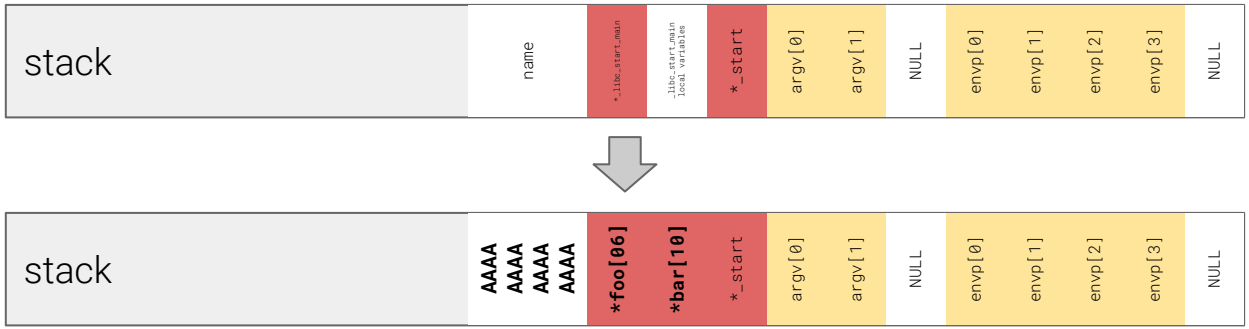


The Stack, with
stored return
addresses

Actions taken by
the program

Conceptual analogy (on source code)

```
01 int main() {
02    char name[16];
03    read(0, name, 128);
04 }
05 int foo() {
06    return open("/flag", 0);
07 }
08 int bar() {
09    int x = open("/notflag", 0);
10    sendfile(1, x, 0, 1024);
11 }
```



ROP by induction

Step 0: overflow the stack

Step n: by controlling the return address, you trigger a *gadget*:

```
0x004005f3: pop rdi ; ret
```

Step n+1: when the gadget returns, *it returns to an address you control* (i.e., the next gadget)

By *chaining* these gadgets, you can perform arbitrary actions in a *ropchain*!

Take-away: ROP is *basically* shellcode

A ROP gadget is equivalent to shellcode, but the instructions available to you are **WEIRD**!

Hacker term: Programming the **Weird Machine** (https://en.wikipedia.org/wiki/Weird_machine), coined in 2009 by Sergey Bratus.

Related concept: accidental turing completeness (http://beza1e1.tuxen.de/articles/accidentally_turing_complete.html).

Fundamentally:

- You get to choose from a set of bizarre meta-instructions already in memory.
- You chain instructions using `ret` (and addresses on the stack).
- *Same lego pieces, new result!*

ROP is FUN!