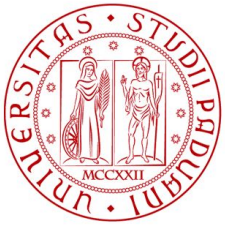
CyberSecurity: Principle and Practice

BSc Degree in Computer Science
2023-2024

Lesson 16: Shellcode

Prof. Mauro Conti
Department of Mathematics
University of Padua
conti@math.unipd.it
http://www.math.unipd.it/~conti/

Teaching Assistants
Tommaso Bianchi
tommaso.bianchi@phd.unipd.it.
Riccardo Preatoni
riccardo.preatoni@studenti.unipd.it



UNIVERSITÀ DEGLI STUDI DI PADOVA

SPRITZ SECURITY & PRIVACY RESEARCH GROUP

DIPARTIMENTO MATEMATICA

Disclaimer

All information presented here has the only purpose of teaching how reverse engineering works.

Use your mad skillz only in CTFs or other situations in which you are legally allowed to do so.

Do not hack the new Playstation. Or maybe do, but be prepared to get legal troubles 😊

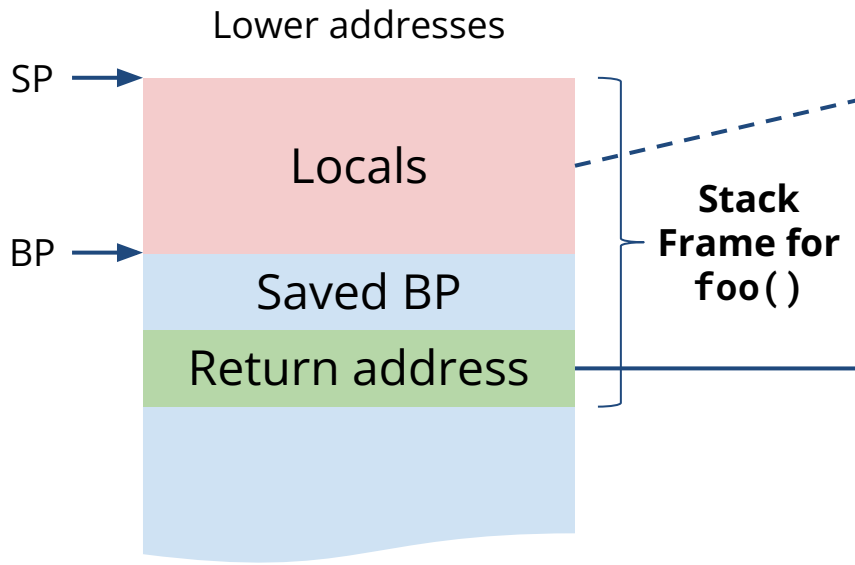
Stack Overflows

The stack contains information that keeps track of the program's control flow.

Overflowing a buffer located on the stack could allows us to hijack the flow to wherever we want.

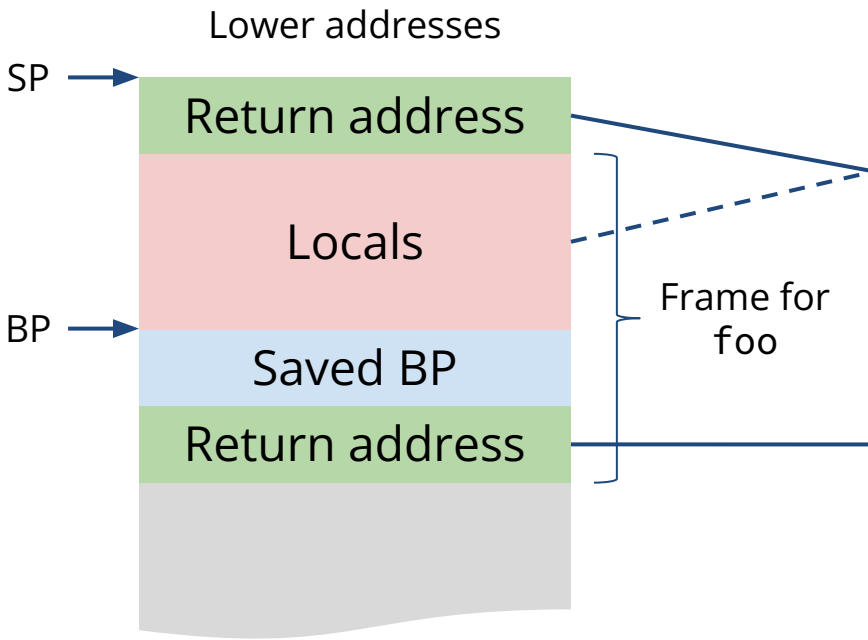
Must read: Aleph One, *Smashing the stack for fun and profit*, Phrack (1996)

The x86 Stack



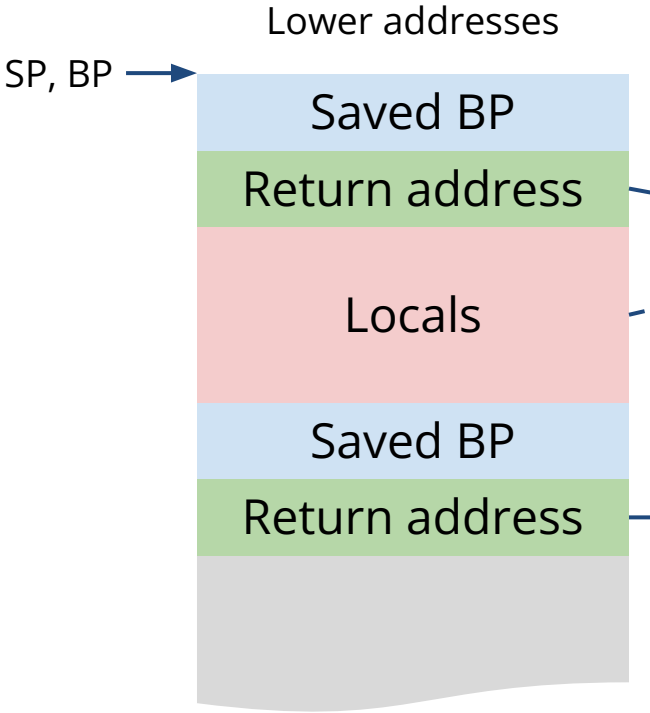

```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

The x86 Stack



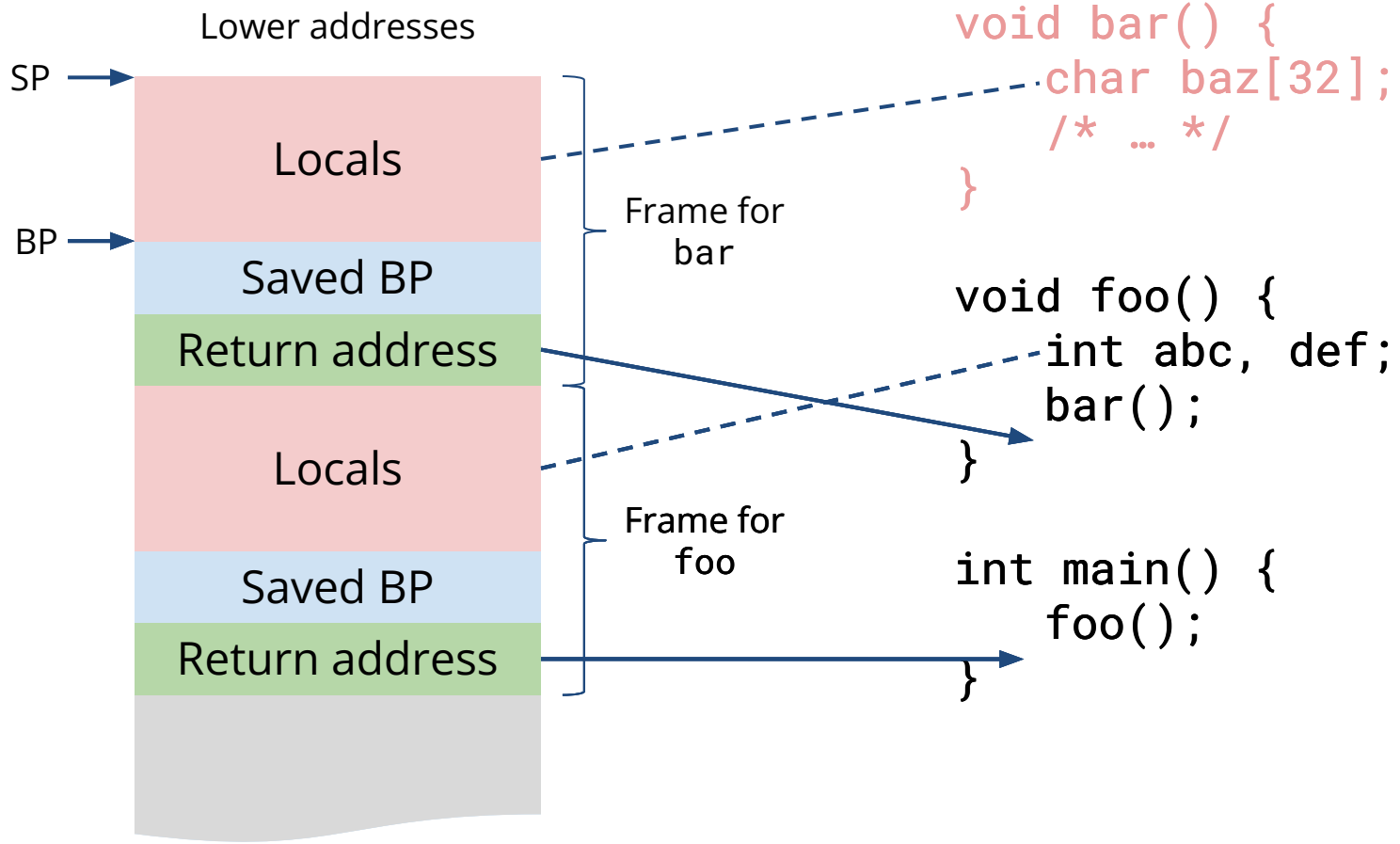

```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

The x86 Stack

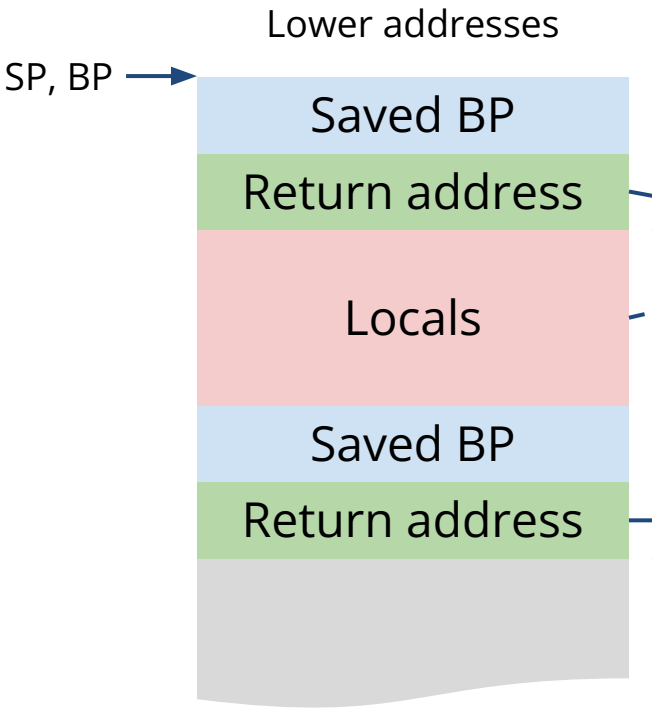



```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

The x86 Stack

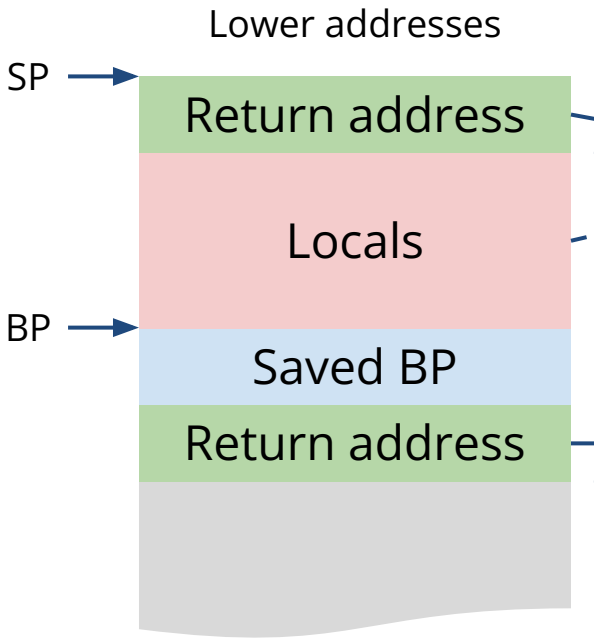


The x86 Stack



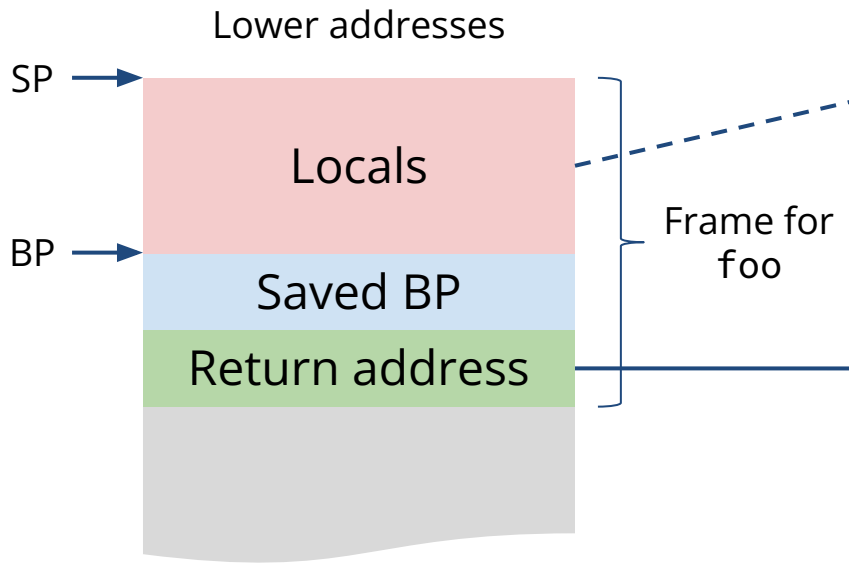

```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

The x86 Stack




```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

The x86 Stack




```
void bar() {
    char baz[32];
    /* … */
}

void foo() {
    int abc, def;
    bar();
}

int main() {
    foo();
}
```

Stack Overflows: an Example!

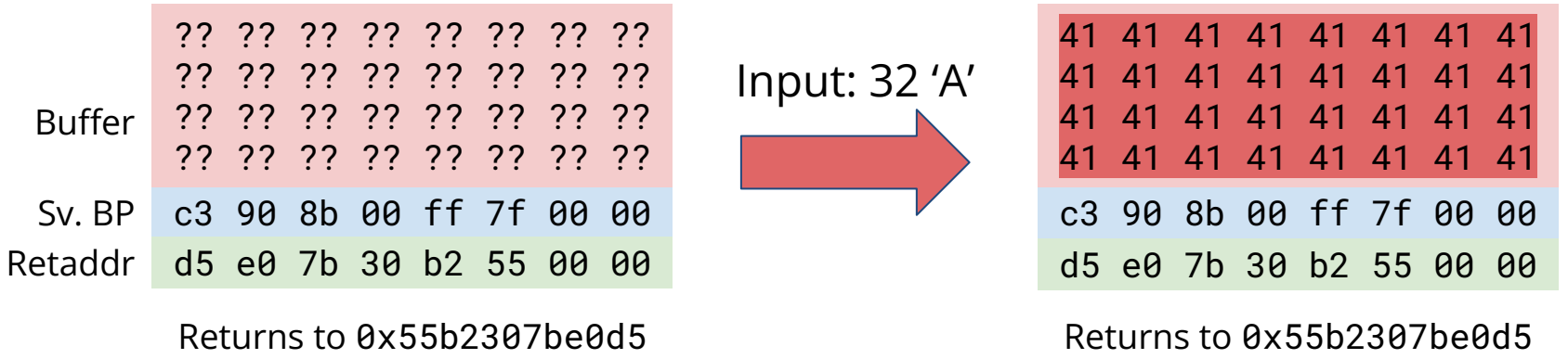
This program copies the user's input to a fixed size 32-byte buffer.

Buffer	?? ??
Sv. BP	c3 90 8b 00 ff 7f 00 00
Retaddr	d5 e0 7b 30 b2 55 00 00

Returns to `0x55b2307be0d5`

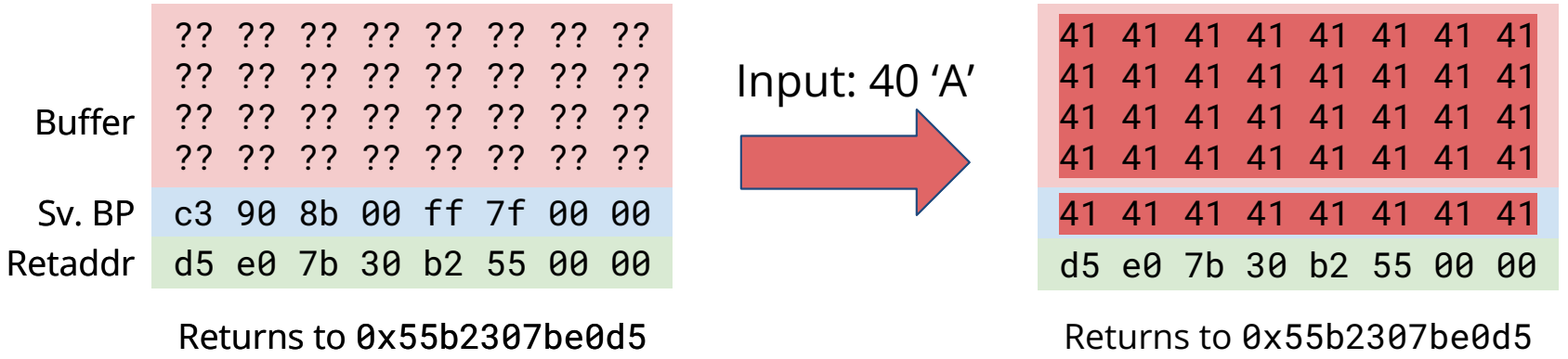
Stack Overflows

This program copies the user's input to a fixed size 32-byte buffer.



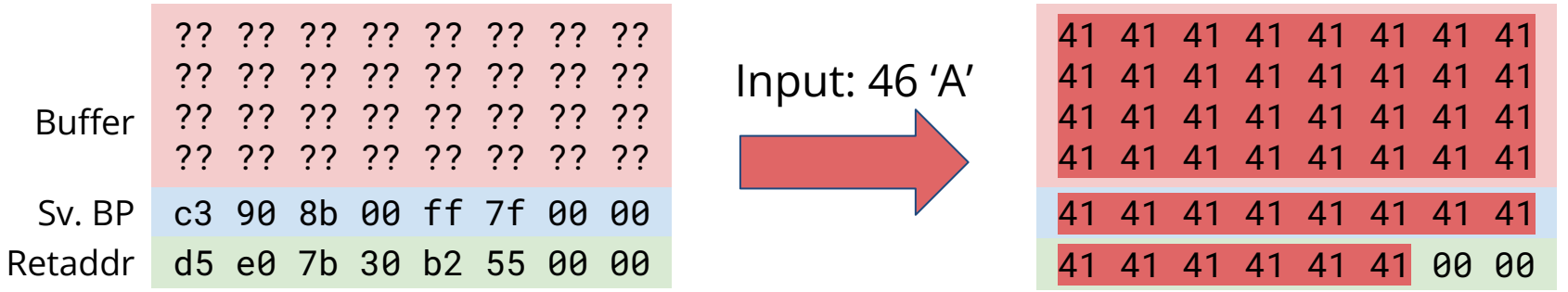
Stack Overflows

This program copies the user's input to a fixed size 32-byte buffer.



Stack Overflows

This program copies the user's input to a fixed size 32-byte buffer.



Shellcode

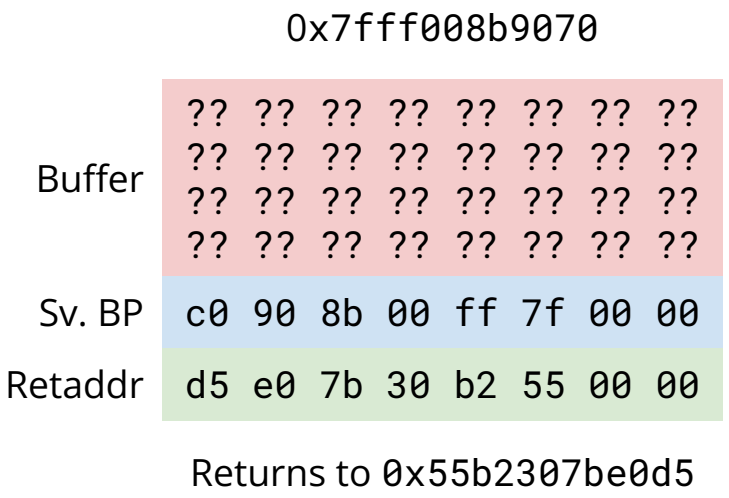
Sometimes there's no "magic" function we can return to.

So let's inject **our own code** into the process.

This code is called *shellcode* because it usually opens a shell.

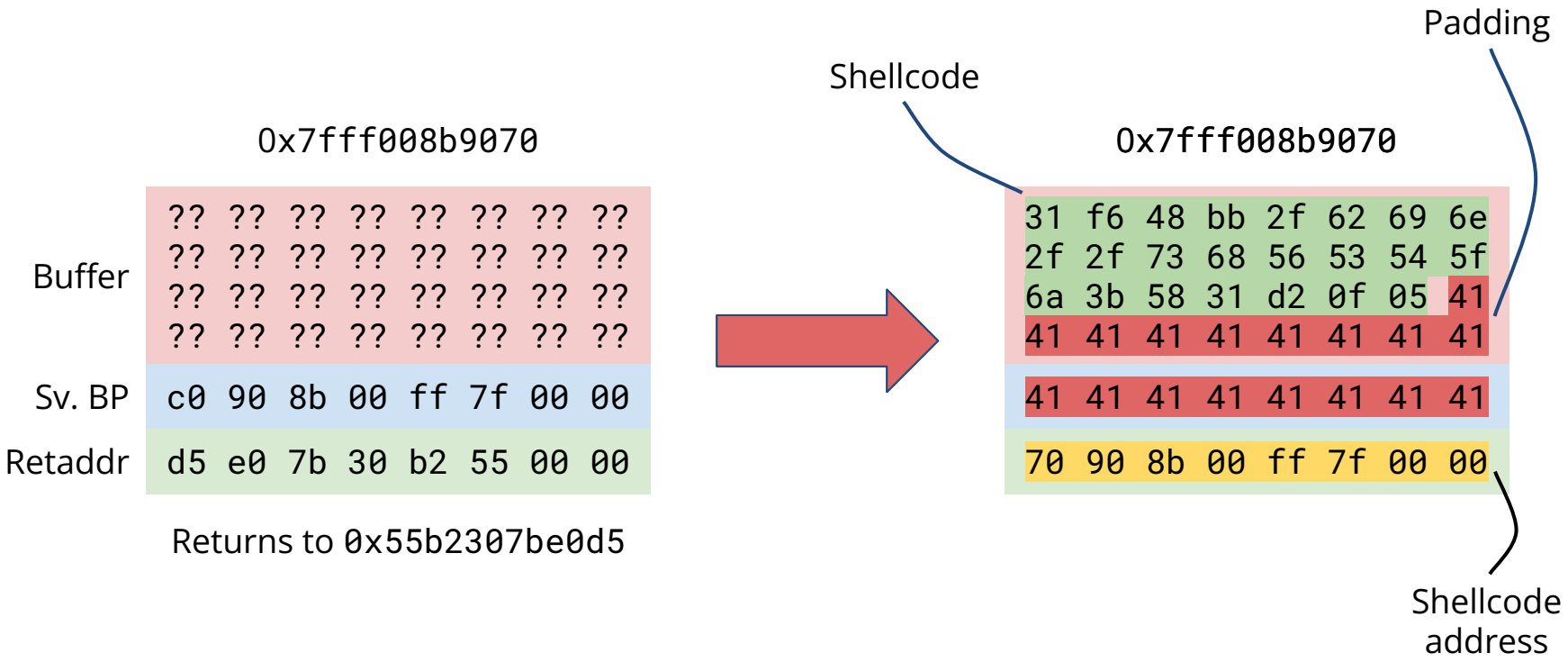
Shellcode

The program copies the user's input to a fixed size 32-byte stack buffer.



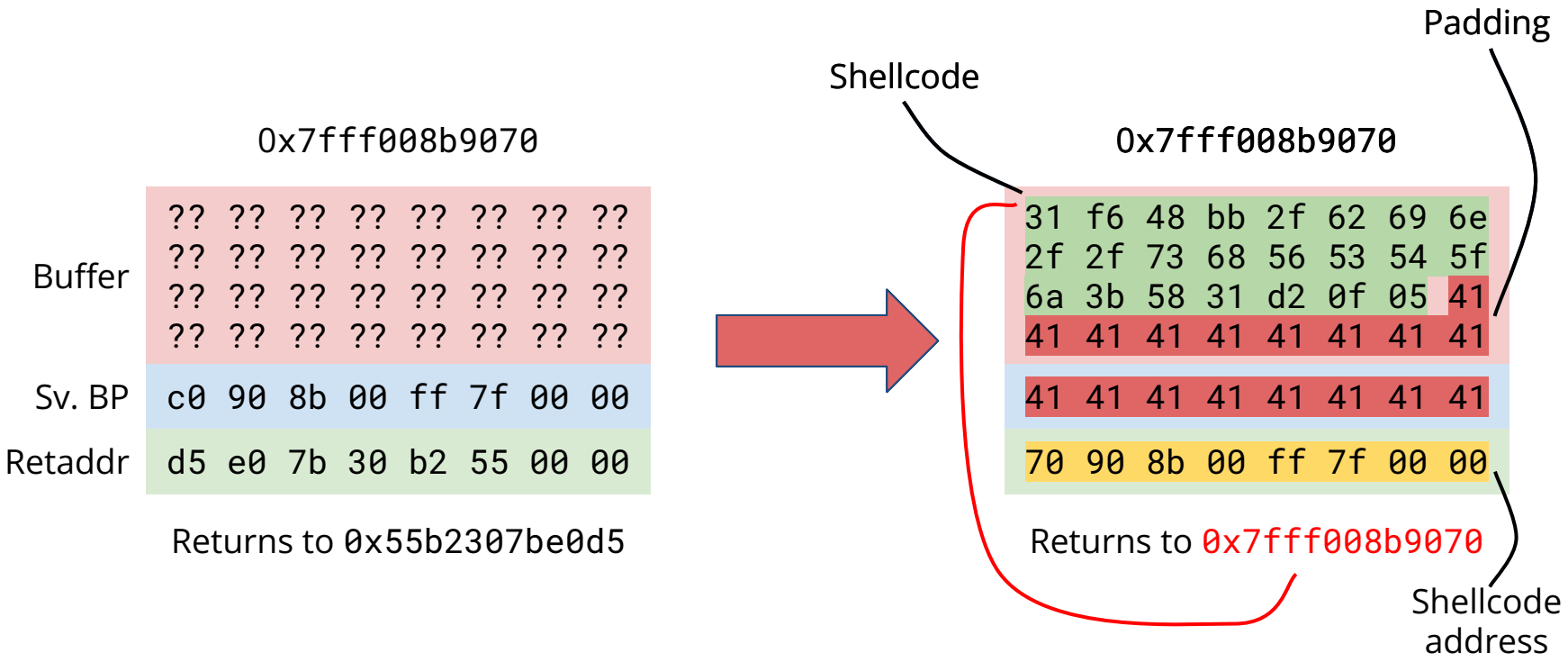
Shellcode

The program copies the user's input to a fixed size 32-byte stack buffer.



Shellcode

The program copies the user's input to a fixed size 32-byte stack buffer.



Mitigation

- Stack Canaries
  - Secret value overwritten by overflow
  - Bypass: infoleak, O(N) bruteforce (forkserver)
- Address Space Layout Randomization (ASLR)
  - Can’t jump if I don’t know where the code is
  - Bypass: infoleak, LSB overwrite, O(N) bruteforce (forkserver)
- Write ⊕ Execute (W⊕X, NX, DEP)
  - Prevent code injection
  - Bypass: code reuse (e.g., ROP)
- Control Flow Integrity (CFI)
  - Restrict control flow transfers to intended paths
  - Bypass: advanced code reuse (e.g., COOP)

Exercises

1) What's your name?
2) Hello mr X, how can we reach the flag?
3) Can you spawn a shell and use that to read the flag.txt?
4) I made a simple shell which allows me to run some specific commands on my server can you test it for bugs?

TIP: Finding the return address from the start of the buffer might be tricky. We can create a pattern that does not repeat itself (cyclic), and see what part of the pattern overwrites the return address. In this way, we can understand the offset (ret_addr - buff_addr)!

Useful gdb-peda commands:

```
pattern_create size [file]
pattern_search
```

Questions? Feedback? Suggestions?