

CyberSecurity: Principle and Practice

*BSc Degree in Computer Science
2021-2022*

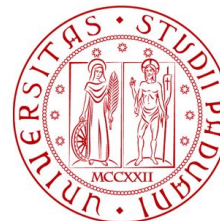
Lesson 15: Shellcode

Prof. Mauro Conti

Department of Mathematics
University of Padua
conti@math.unipd.it
<http://www.math.unipd.it/~conti/>

Teaching Assistants

Luca Pajola
pajola@math.unipd.it.
Pier Paolo Tricomi
pierpaolo.tricomi@phd.unipd.it



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP



DIPARTIMENTO 1
MATEMATICA

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 😊

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

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

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

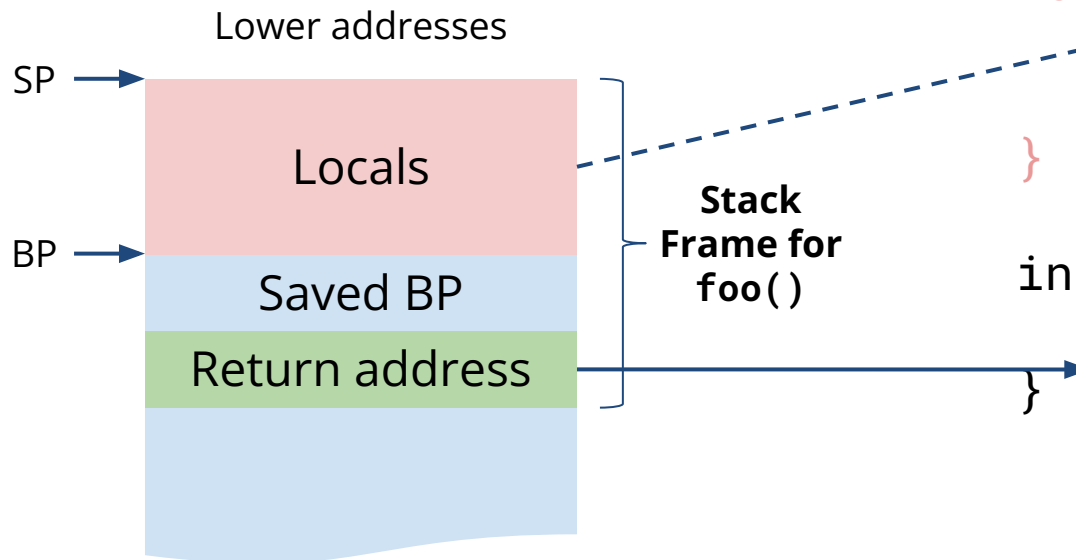
The x86 Stack



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

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

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



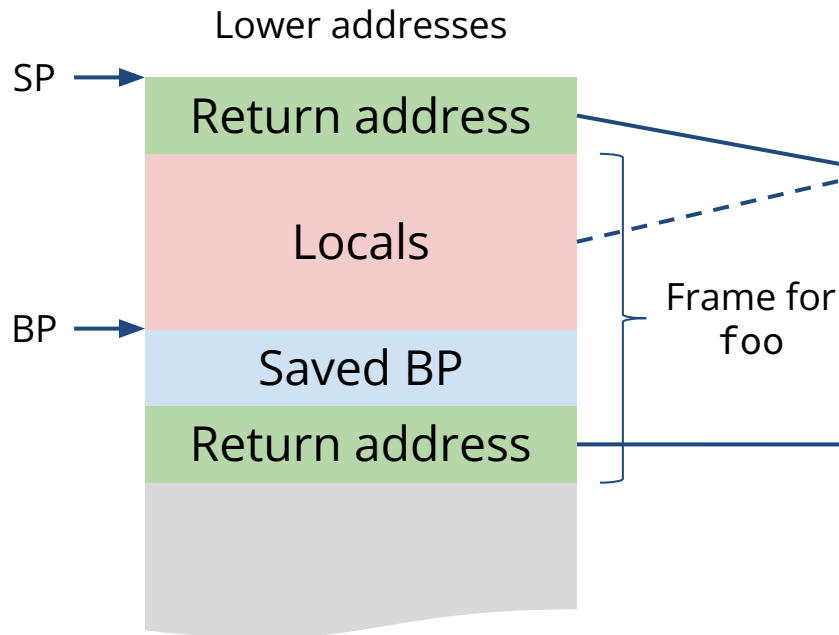
The x86 Stack



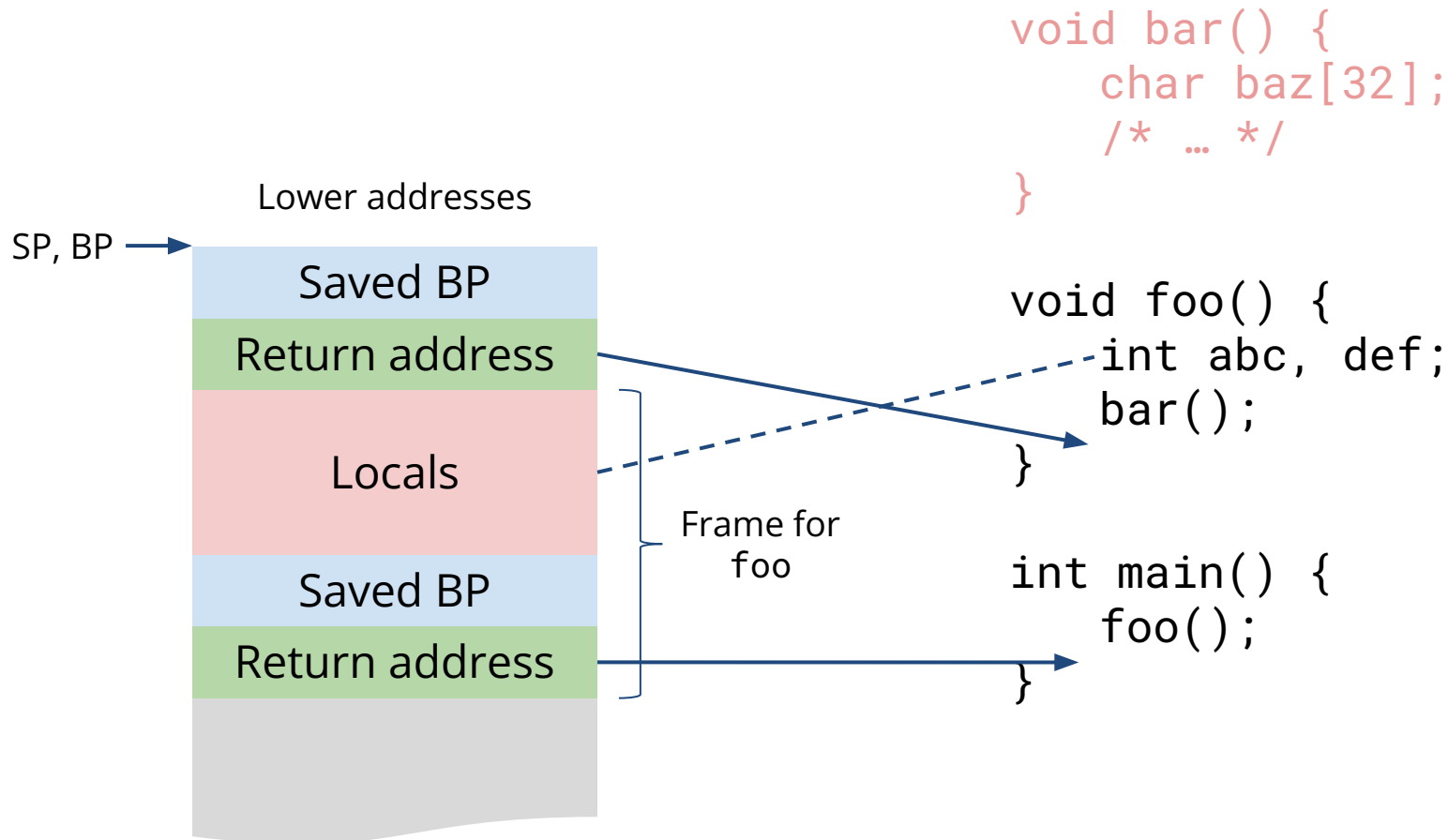
```
void bar() {  
    char baz[32];  
    /* ... */  
}
```

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

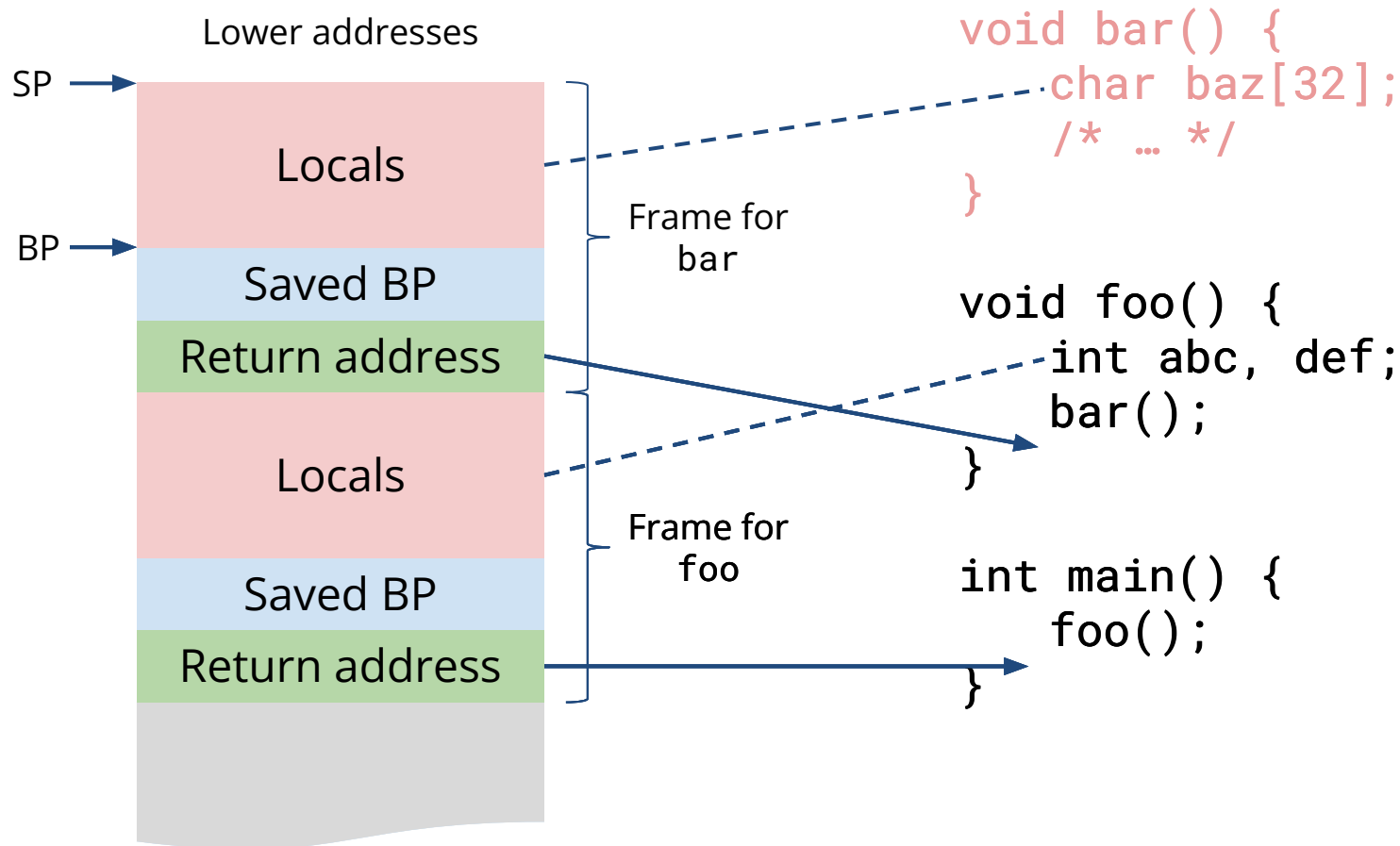
```
int main() {  
    foo();  
}
```



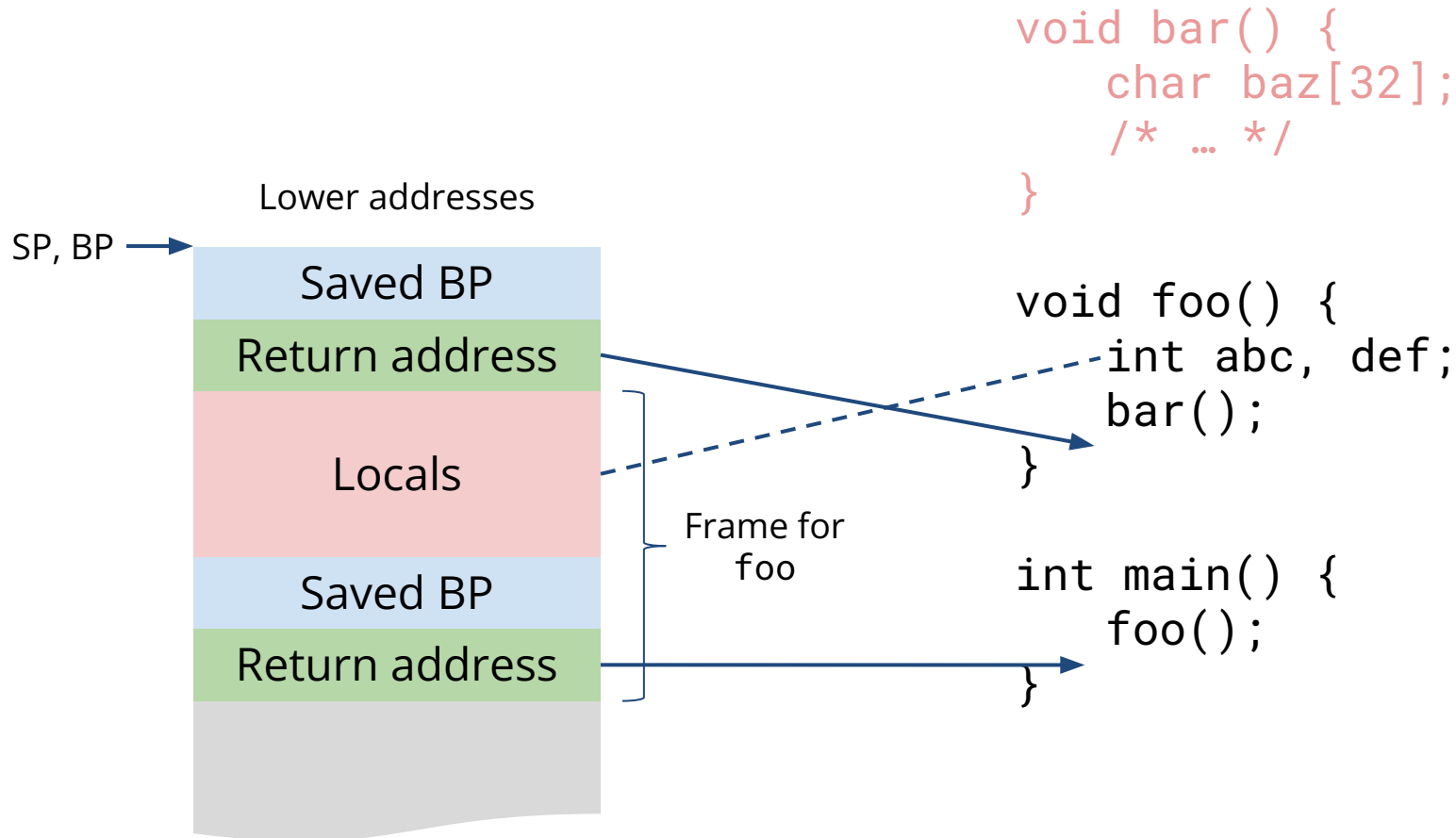
The x86 Stack



The x86 Stack



The x86 Stack



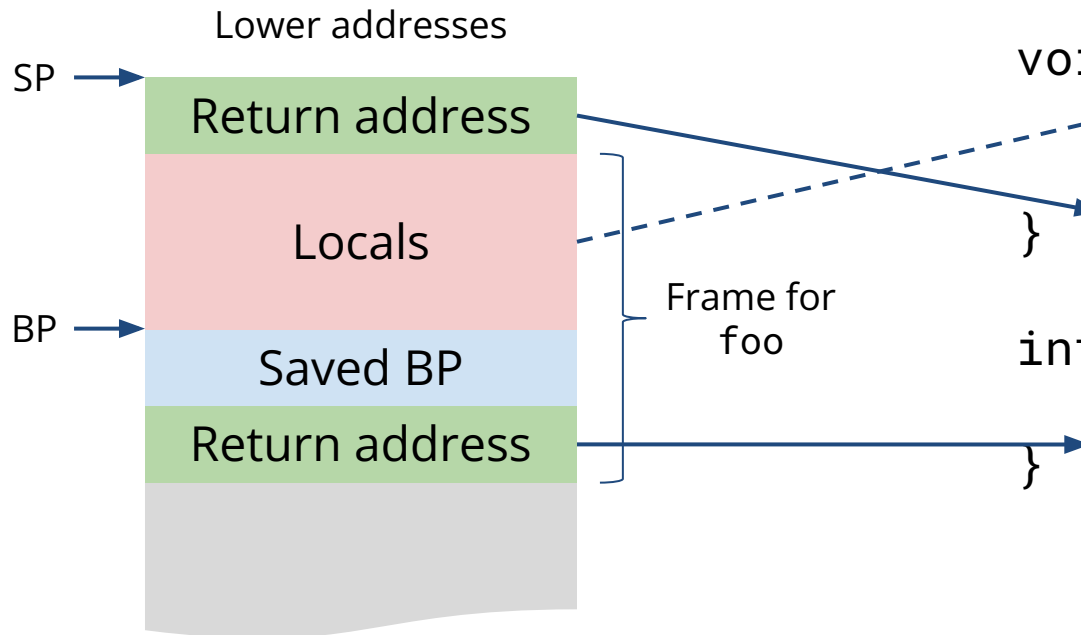
The x86 Stack



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

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

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



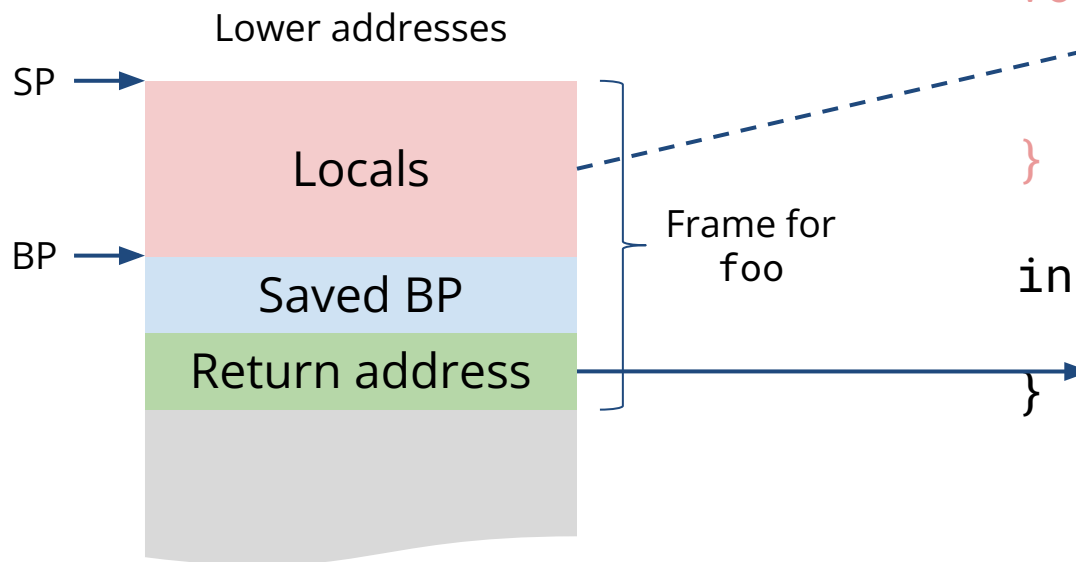
The x86 Stack



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

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

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



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

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

Input: 32 'A'



41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
c3	90	8b	00	ff	7f	00	00	00
d5	e0	7b	30	b2	55	00	00	00

Returns to 0x55b2307be0d5

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

Input: 40 'A'



41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
41	41	41	41	41	41	41	41	41
d5	e0	7b	30	b2	55	00	00	00

Returns to 0x55b2307be0d5

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.

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

0x7fff008b9070

Buffer

??	??	??	??	??	??	??	??
??	??	??	??	??	??	??	??
??	??	??	??	??	??	??	??
??	??	??	??	??	??	??	??

Sv. BP

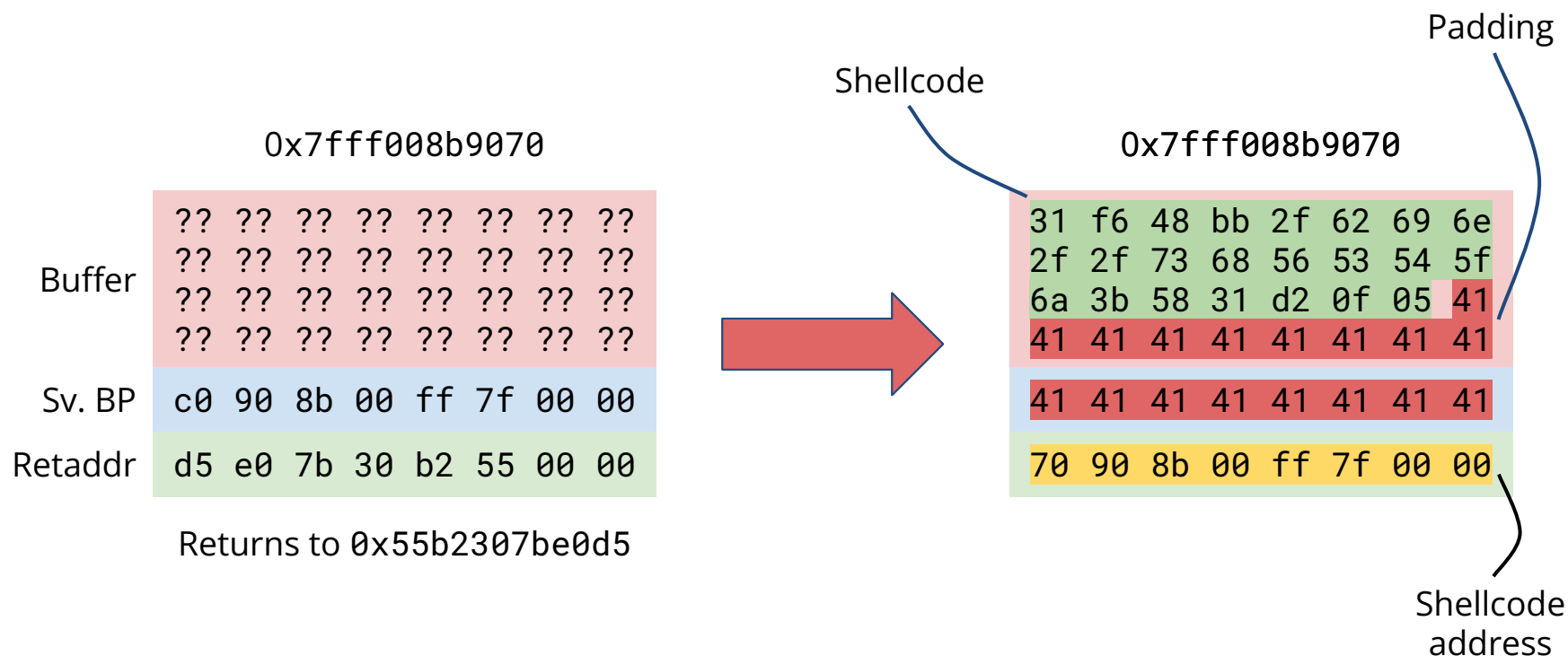
c0	90	8b	00	ff	7f	00	00
----	----	----	----	----	----	----	----

Retaddr

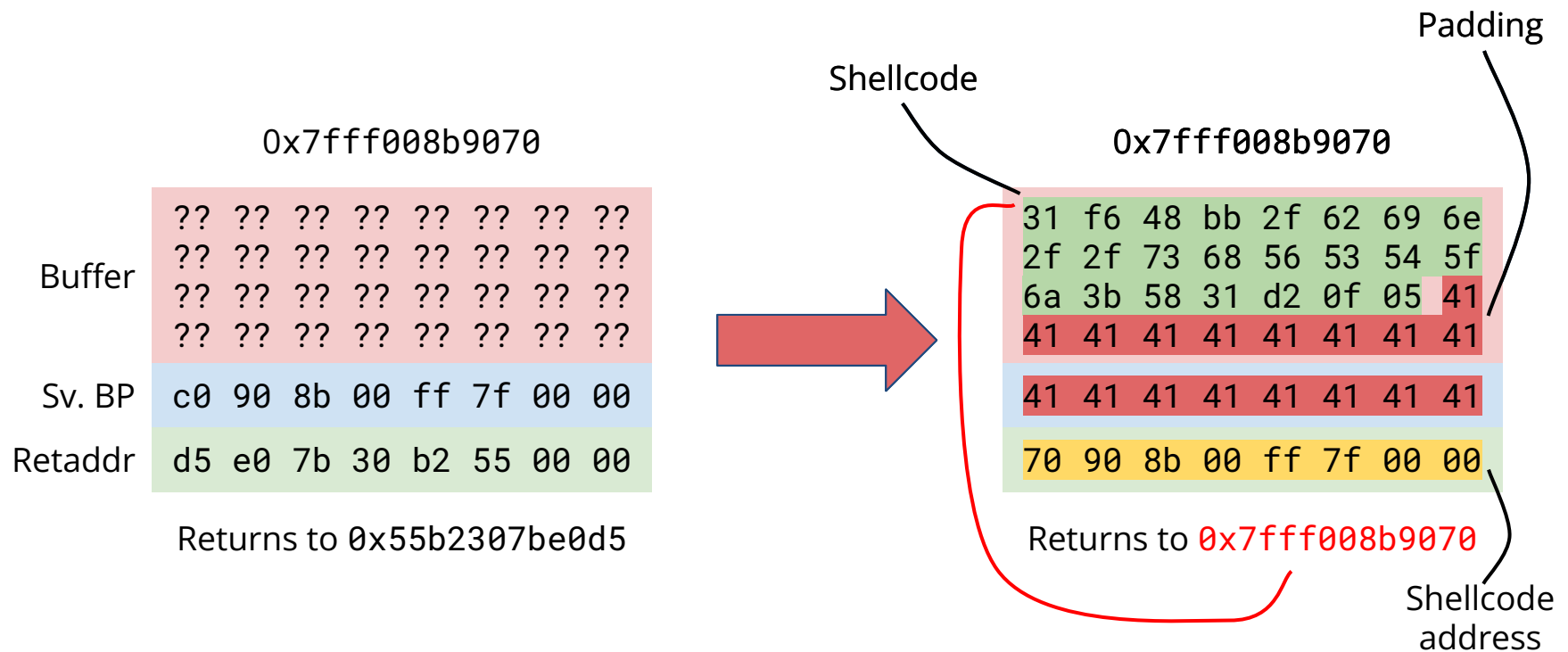
d5	e0	7b	30	b2	55	00	00
----	----	----	----	----	----	----	----

Returns to 0x55b2307be0d5

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



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



- 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 \oplus Execute ($W\oplus X$, 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)

- 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?



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

