

CyberSecurity: Principle and Practice

*BSc Degree in Computer Science
2021-2022*

Lesson 12: Patching

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



DIPARTIMENTO¹
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 😊

Patching

Why do I need it?

You may need to fix binary bugged

You may need to change the behavior of the binary

You may not have the source code, so binary is the only way

Patching

Change the program through static analysis

We understand what the binary does, and change its bytes for our purposes

Patching != instrumentation

How it is supposed to work

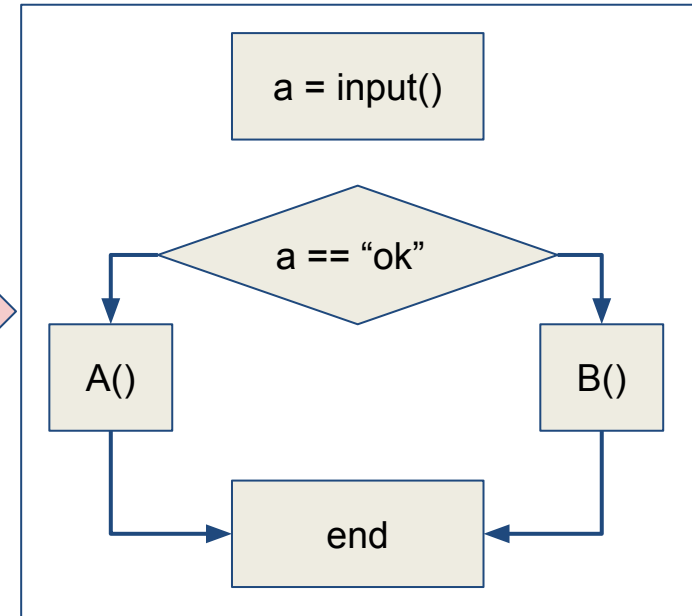
Reversing

00101010110

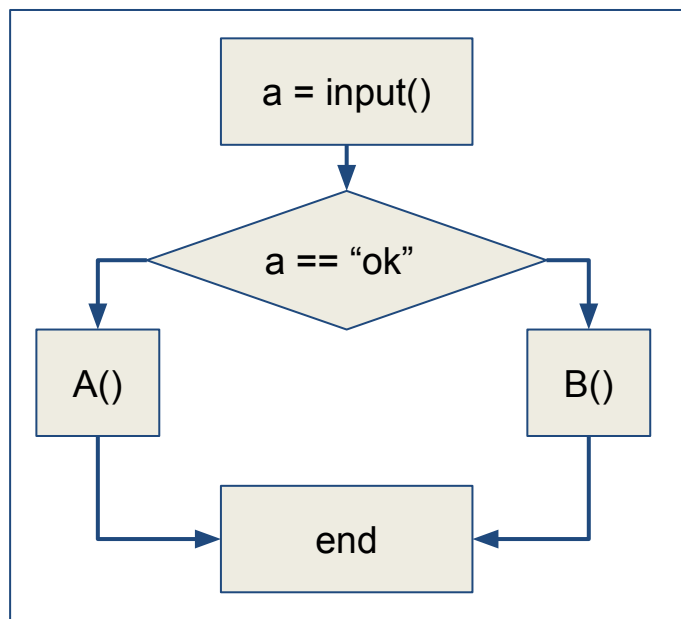
to_crack.exe



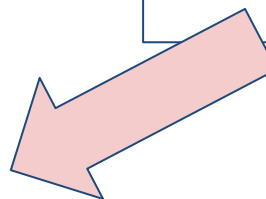
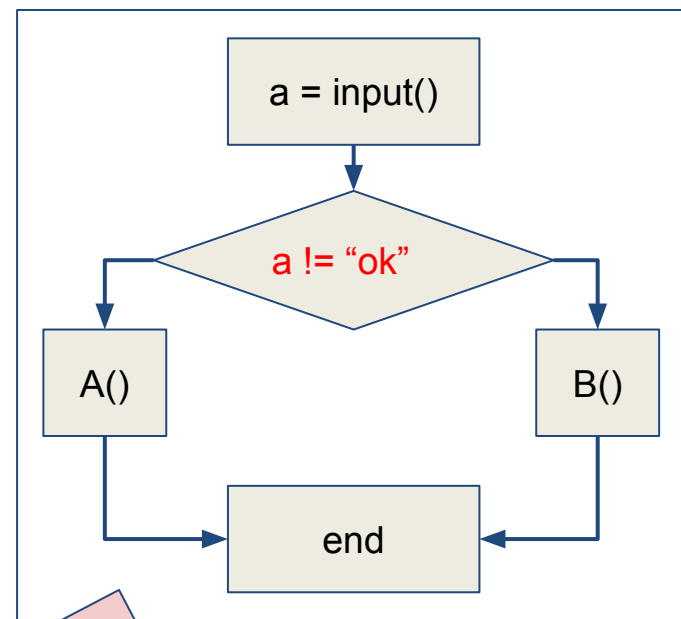
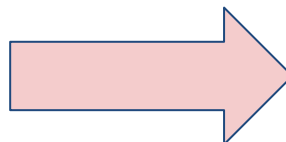
You



How it is supposed to work



Patch



00101010110

to_crack_but_patched.exe

Basically, all the static analyzers saw till now:

- IDA Pro ← \$\$\$\$\$\$ (free version doesn't work well)
- Binja ← \$\$\$
- Ghidra ← free, but doesn't work
- Hex Editor ← Free and reasonable ok (for simple things)
- Radare2 ← Free and reasonable ok (for simple things)

1. Make a copy of the binary
2. Use a disassembler to find the instruction(s) to patch
3. Look at the hex bytes and find them in the binary
4. Edit the bytes and save
5. Run the patched binary and pray

Opcodes x86: <http://ref.x86asm.net/coder64.html>

Find Hex Bytes in the binary



Easy way: Copy consecutive bytes from disassembler Hex View around the instruction to patch and search for them in the Binary using the Hex Editor. Then locate the exact byte(s) to patch.

Be sure that Hex Editor and Disassembler share the same view (type of Endian, number of grouped bytes in the view...)

Little Endian Single grouped
0x11 0x22 0x33 0x44

Little Endian Double grouped
0x2211 0x4433

Hard way: calculate the Relative Virtual Address ([Learn Here](#))

PLEASE NOTE: Always compare a bunch of bytes (hex editor vs disassembler) to see if you found the right area

1. Make a copy of the binary
2. Open the binary in write mode
r2 -w <binary>
3. Launch the analysis
aaaaa
4. Seek for the wanted function
s <function name>
5. Print Decompiled Function
pdf
6. Understand what to patch
“use brain”
7. Go to the address to patch
s <address>
8. Patch using wa instruction
9. Double check with pdf

```
[0x000044e7]> wa?
Usage: wa[of*] [arg]
| wa nop          write nopcode using asm.arch and asm.bits
| wai jmp 0x8080   write inside this op (fill with nops or error if doesnt fit)
| wa* mov eax, 33  show 'wx' op with hexpair bytes of assembled opcode
| "wa nop;nop"     assemble more than one instruction (note the quotes)
| waf f.asm        assemble file and write bytes
| wao?            show help for assembler operation on current opcode (hack)
[0x000044e7]> "wa nop;nop;nop;nop;nop;nop;nop"
Written 7 byte(s) (nop;nop;nop;nop;nop;nop;nop) = wx 90909090909090
[0x000044e7]> 
```

- fill with NOP
- invert branches (JE <-> JNE)
- remove branches (NOP/JMP)
- change some constant (e.g., strings/number)
- paste new functions (not so common, but can happen)

Questions? Feedback? Suggestions?



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

