Reversing Lab 01

Carlo Ramponi <carlo.ramponi@unitn.it>

Approaching Reverse Challenges

Carlo Ramponi

Approaching Reverse Challenges

When approaching a Reverse Engineering Challenge, before jumping straight into Ghidra, you firstly need to understand what kind of file you are dealing with.

Useful tools:

- `file` - determine file type
- `strings` - print the sequences of printable characters in files
- `hexdump` - display file contents in hexadecimal, decimal, octal, or ascii
- The binary itself! If you can execute it (through an emulator, perhaps), do it!
- `ltrace` - A library call tracer
- `strace` - trace system calls and signals

Approaching Reverse Challenges

When you need to **start reversing**, import the file in **`Ghidra`**, depending on the file format, you might need to **instruct** **`Ghidra`** on how to open the file, by **installing an extension** perhaps.

Common binary formats, such as **`ELF`** or **`PE`**, are straightforward, but you might encounter some strange files, trust me!

Approaching Reverse Challenges

When in **`Ghidra`**, you need to **identify the interesting code**

- If there are symbols, look at the function names 😅
- Look for the **entry address**
  - This is where the program starts executing
  - For ELF or PE binaries this is straightforward, other formats might require a little bit of googling.
- Look where interesting **library functions** (e.g. **`system`**) are used (i.e. **`XREFS`**)
- Look for **interesting strings** and their **`XREFS`**

Approaching Reverse Challenges

When you've identified the functions you need to reverse:

- Don't just look at the **decompiler**:
  - The decompiler is **not perfect**, it could have missed something
  - The disassembly should be your ground truth of what the program does
- Reverse engineering requires **manual work**:
  - **Rename variables** and functions
  - **Retype variables** and function arguments
  - **Create complex types** in Ghidra (structures, classes, …)
- Google is your friend!

Challenges

Chall 00 - Reversing 101

Description

This is a simple reversing challenge.
You have to find the flag in the binary.

Points: *100*
Author: *carlo*
Attachments: *chall* (binary file)

Solution:
Chall 00 - Reversing 101

Carlo Ramponi

Chall 01 - Reversing 102

Description

The first one was too easy?
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *100*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:
1. <REDACTED>
2. <REDACTED>

Chall 01 - Reversing 102

Description

The first one was too easy?
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *100*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:
1. The function to be reversed is `check_flag`
2. <REDACTED>

Chall 01 - Reversing 102

Description

The first one was too easy?
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *100*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:
1. The function to be reversed is `check_flag`
2. The **input is XORed** with its length and then compared to a fixed array

Solution:
Chall 01 - Reversing 102

Carlo Ramponi

Chall 02 - Reversing 103

Description

I really thought that XOR was a good encryption algorithm,
but it seems that you have found the flag in the binary.
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *200*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:
1. <REDACTED>
2. <REDACTED>

Chall 02 - Reversing 103

Description

I really thought that XOR was a good encryption algorithm,
but it seems that you have found the flag in the binary.
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *200*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. **Stripped binary**: look for the **entry address** and look at **`__libc_start_main`**
2. <REDACTED>

Chall 02 - Reversing 103

Description

I really thought that XOR was a good encryption algorithm,
but it seems that you have found the flag in the binary.
No problem, this time our engineers have used a more advanced encryption algorithm.

Points: *200*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. **Stripped binary**: look for the **entry address** and look at **`__libc_start_main`**
2. **AES** is used, identify the elements (**key, IV, ciphertext, mode**) and decrypt it

Solution:
Chall 02 - Reversing 103

Carlo Ramponi

Chall 03 - Reversing 105

Description

Pretty simple reversing challenge, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. <REDACTED>
2. <REDACTED>
3. <REDACTED>
4. <REDACTED>

Chall 03 - Reversing 105

Description

Pretty simple reversing challenge, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. Find the main function and **carefully understand** what each sub-function does
2. <REDACTED>
3. <REDACTED>
4. <REDACTED>

Chall 03 - Reversing 105

Description

Pretty simple reversing challenge, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. Find the main function and **carefully understand** what each sub-function does
2. Looks like the function at `0x101361` does not contain valid x86 code, strange…
3. <REDACTED>
4. <REDACTED>

Chall 03 - Reversing 105

Description

Pretty simple reversing challenge, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. Find the main function and **carefully understand** what each sub-function does
2. Looks like the function at `0x101361` does not contain valid x86 code, strange…
3. The func at `0x1011a9` does something with `mprotect` and the **strange func**
4. <REDACTED>

Chall 03 - Reversing 105

Description

Pretty simple reversing challenge, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. Find the main function and **carefully understand** what each sub-function does
2. Looks like the function at `0x101361` does not contain valid x86 code, strange…
3. The func at `0x1011a9` does something with `mprotect` and the **strange func**
4. The code is **XORed** before being executed, can you do the same?

Solution:
Chall 03 - Reversing 105

Carlo Ramponi

Chall 04 - Reversing++

Description

Let's add some ++ to the reversing challenges!

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. <REDACTED>
2. <REDACTED>
3. <REDACTED>

Chall 04 - Reversing++

Description

Let's add some ++ to the reversing challenges!

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. The challenge is written in **C++**, what's the **main feature of an OOP language**?
2. <REDACTED>
3. <REDACTED>

Chall 04 - Reversing++

Description

Let's add some ++ to the reversing challenges!

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. The challenge is written in **C++**, what's the **main feature of an OOP language**?
2. **Virtual Methods**? What are they? How are they **implemented** in the binary?
3. <REDACTED>

Chall 04 - Reversing++

Description

Let's add some ++ to the reversing challenges!

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)
Hints:

1. The challenge is written in **C++**, what's the **main feature of an OOP language**?
2. **Virtual Methods**? What are they? How are they **implemented** in the binary?
3. You should really look at the `FlagChecker` class...

Solution:
Chall 04 - Reversing++

Carlo Ramponi

Rated Challenge - The x86 police

Description

The program isn't doing anything illegal, innit?

Points: *300*
Author: *carlo*
Attachments: *chall* (binary file)

Deadline: April 23th, 2024 at 23:59

GL HF!

Carlo Ramponi