Binary Exploitation Lab 01

Carlo Ramponi <carlo.ramponi@unitn.it>

Approaching `pwn` Challenges

Carlo Ramponi

Approaching pwn Challenges

When approaching a pwn Challenge, there are a few tools you can use to help you understand what you can and can’t do!

Useful tools:

- `file` - determine file type
- `strings` - print the sequences of printable characters in files
- `pwn checksec` (from **pwntools**) - shows the security measures included in a binary file, such as DEP, PIE, Stack Canaries, …

Approaching pwn Challenges

In many pwn challenges, you'll need to compute an offset, e.g. in buffer overflows you want to know the distance between the buffer and the memory location you want to overwrite.

Even though this is statically computable, it is often easier to inject something and look at the result, here is where a pattern string comes handy:

```
$ pwn cyclic N
$ pwn cyclic --lookup=STRING/NUMBER
```

```
gdb> pattern create N
gdb> pattern offset STRING/NUMBER
```

Approaching pwn Challenges

By applying reverse engineering techniques or by looking at the source code (if it is available), you should be able to **identify the vulnerabilities**, e.g.:

- **Buffer overflows**: pay attention to where buffers are initialized, copied, …
- **Format String Vulnerabilities**: pay attention to f-functions calls
- …

Approaching pwn Challenges

After **identifying the vulnerability**, reason about what you can do with it, and find a way to exploit it to get the flag.

If there is nothing in the binary that prints the flag you'll probably have to **spawn a shell** (e.g. `system(“/bin/bash”)`) and `cat` it.

Challenges

Chall 00 - Coffee Machine

Description

Be careful not to spill your coffee on the floor,
it's a mess to clean up.

Points: *100*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Solution:

Chall 00 - Coffee Machine

Chall 01 - Time Machine

Description

Time travel is a dangerous business, you never know what you might find.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. <REDACTED>
2. <REDACTED>

Chall 01 - Time Machine

Description

Time travel is a dangerous business, you never know what you might find.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. **Where** is **check**? Use a gdebugger and a **pattern** string
2. <REDACTED>

Chall 01 - Time Machine

Description

Time travel is a dangerous business, you never know what you might find.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. **Where** is **check**? Use a $_g$debugger and a **pattern** string
2. How is an **int** variable stored in memory? Encoding, endianness, …

Solution:

Chall 01 - Time Machine

Chall 02 - Teleport

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. <REDACTED>
2. <REDACTED>

Chall 02 - Teleport

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. What can we overwrite now? Where’s the code we want to execute?
2. <REDACTED>

Chall 02 - Teleport

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. What can we overwrite now? Where's the code we want to execute?
2. Just like in the prev chall, you have to ow a value somewhere on the stack

Solution:

Chall 02 - Teleport

Chall 03 - Teleport 2.0

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. <REDACTED>
2. <REDACTED>

Chall 03 - Teleport 2.0

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. No `win` function this time, you might wanna run `pwn checksec` on the binary
2. <REDACTED>

Chall 03 - Teleport 2.0

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. No `win` function this time, you might wanna run `pwn checksec` on the binary
2. If there is no `win` function, you might try to inject one!

Solution:

Chall 03 - Teleport 2.0

Chall 04 - Teleport 2.1

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. <REDACTED>
2. <REDACTED>

Chall 04 - Teleport 2.1

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. There is **no leak** of the stack address this time, can you find something **useful**?
2. <REDACTED>

Chall 04 - Teleport 2.1

Description

This teleporter can send you anywhere in the world, but it's not very stable.

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. There is **no leak** of the stack address this time, can you find something **useful**?
2. Where are you going to place that **`shellcode`**? You want to execute it!

Solution:

Chall 04 - Teleport 2.1

Chall 05 - Teleport 3.0

Description
This teleporter can send you anywhere in the world, but it's not very stable.
We've been told that previous versions had some security issues,
but we're sure we've fixed them all. Can you prove us wrong?

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

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

Chall 05 - Teleport 3.0

Description
This teleporter can send you anywhere in the world, but it's not very stable.
We've been told that previous versions had some security issues,
but we're sure we've fixed them all. Can you prove us wrong?

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. Stack **canary** has been enabled, how can we kill it?
2. <REDACTED>
3. <REDACTED>

Chall 05 - Teleport 3.0

Description

This teleporter can send you anywhere in the world, but it's not very stable.
We've been told that previous versions had some security issues,
but we're sure we've fixed them all. Can you prove us wrong?

Points: *200*

Author: *carlo*

Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. Stack **canary** has been enabled, how can we kill it?
2. You can try to **leak** the canary somehow!
3. <REDACTED>

Chall 05 - Teleport 3.0

Description
This teleporter can send you anywhere in the world, but it's not very stable.
We've been told that previous versions had some security issues,
but we're sure we've fixed them all. Can you prove us wrong?

Points: *200*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Hints:

1. Stack **canary** has been enabled, how can we kill it?
2. You can try to **leak** the canary somehow!
3. The canary value always starts with a **null-byte**, what if we **overwrite** it?

Solution:

Chall 05 - Teleport 3.0

Rated Challenge:
BASH - Basic Asynchronous Shell

Description

Take a look at my new shell!
It leverages the power of linux's multiplexing capabilities to provide a fast and responsive shell experience.

Points: *500*
Author: *carlo*
Attachments:

- *bin* (binary file)
- *chall.c* (source code)

Deadline: Tuesday, 14th of May at 23:59

GL HF!

That's All
Folks