Binary Exploitation Lab 02

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:
True Random Password Generator (TRPG)

Description

Well, the name says it all. This is a True Random Password Generator (TRPG).

Points: *100*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. <REDACTED>

Chall 00:
True Random Password Generator (TRPG)

Description

Well, the name says it all. This is a True Random Password Generator (TRPG).

Points: *100*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. Unlike previous versions, the password is really **unpredictable**, but maybe we can **leak** it!

Solution:
Chall 00
True Random Password Generator (TRPG)

Chall 01:
True Real Random Password Generator (TRRPG)

Description
We, at UniTN, learned from our mistakes and now we have a True Real Random Password Generator (TRRPG).

Points: *100*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. <REDACTED>

Chall 01:
True Real Random Password Generator (TRRPG)

Description
We, at UniTN, learned from our mistakes and now we have a True Real Random Password Generator (TRRPG).

Points: *100*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. The `parameter$` field of a format parameter might help you!

Solution:
Chall 01
True Real Random Password Generator (TRRPG)

Chall 02:
Very True Real Random Password Generator (VTRRPG)

Description
I guess the stack is not a safe place for secrets, let's move it elsewhere.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. <REDACTED>
2. <REDACTED>

Chall 02:
Very True Real Random Password Generator (VTRRPG)

Description
I guess the stack is not a safe place for secrets, let's move it elsewhere.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. The secret is **not on the stack**, but there's its **pointer**, can we use it?
2. <REDACTED>

Chall 02:
Very True Real Random Password Generator (VTRRPG)

Description
I guess the stack is not a safe place for secrets, let's move it elsewhere.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. The secret is **not on the stack**, but there's its **pointer**, can we use it?
2. Isn't there a format parameter that **`dereferences`** pointers?

Solution:

Chall 02

Very True Real Random Password Generator (VTRRPG)

Chall 03 - Not A Password Generator

Description
I see, you can guess everything, don't you?
Try to guess this one.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. <REDACTED>
2. <REDACTED>

Chall 03 - Not A Password Generator

Description
I see, you can guess everything, don't you?
Try to guess this one.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. What's the only format parameter that **writes** to memory?
2. <REDACTED>

Chall 03 - Not A Password Generator

Description
I see, you can guess everything, don't you?
Try to guess this one.

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. What's the only format parameter that **writes** to memory?
2. **%n** needs a **pointer** to know where to write, how can we put it on the **stack**?

Solution:

Chall 03

Not A Password Generator

Chall 04 - Not (YET) A Password Generator

Description
I guess you really know your way around format strings, how about this one?

Points: *300*
Author: *carlo*
Attachments:

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

Hints:

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

Chall 04 - Not (YET) A Password Generator

Description
I guess you really know your way around format strings, how about this one?

Points: *300*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. This time you need to write a **specific value**, padding might be handy…
2. <REDACTED>
3. <REDACTED>

Chall 04 - Not (YET) A Password Generator

Description
I guess you really know your way around format strings, how about this one?

Points: *300*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. This time you need to write a **specific value**, padding might be handy…
2. Padding is too **big**?, maybe **write smaller values**, one by one…
3. <REDACTED>

Chall 04 - Not (YET) A Password Generator

Description
I guess you really know your way around format strings, how about this one?

Points: *300*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)

Hints:
1. This time you need to write a **specific value**, padding might be handy…
2. Padding is too **big**?, maybe **write smaller values**, one by one…
3. What about the **size specifier** in format parameters? **%hhn**!

Solution:

Chall 04

Not (YET) A Password Generator

Chall 05 - LIBC playground

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

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

- *bin* (binary file)
- *libc.so.6* (shared library)

Hints:

1. <REDACTED>

Chall 05 - LIBC playground

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

Points: *100*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *libc.so.6* (shared library)

Hints:
1. Thanks to **PIE** and **ASLR**, we can't know where **`libc`** will be loaded in memory, but we sure know the "***distance***" between objects it contains!

Solution:
Chall 05
LIBC playground

Chall 06 - LIBC playground 2.0

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)
- *libc.so.6* (shared library)

Hints:
1. <REDACTED>
2. <REDACTED>
3. <REDACTED>

Chall 06 - LIBC playground 2.0

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

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

- *bin* (binary file)
- *chall.c* (source code)
- *libc.so.6* (shared library)

Hints:

1. It's not enough to "*guess*" **`system`**'s address, maybe there's a **`BoF`**?
2. <REDACTED>
3. <REDACTED>

Chall 06 - LIBC playground 2.0

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)
- *libc.so.6* (shared library)

Hints:
1. It's not enough to "*guess*" **`system`**'s address, maybe there's a **`BoF`**?
2. When **`exit`** is called, the function **does not return**, you don't want that!
3. <REDACTED>

Chall 06 - LIBC playground 2.0

Description
Let's play with the LIBC and dynamic linking! Can you exploit this binary?

Points: *200*
Author: *carlo*
Attachments:
- *bin* (binary file)
- *chall.c* (source code)
- *libc.so.6* (shared library)

Hints:
1. It's not enough to "*guess*" **`system`**'s address, maybe there's a **`BoF`**?
2. When **`exit`** is called, the function **does not return**, you don't want that!
3. A **`one_gadget`** might be very useful here…

Solution:

Chall 06

LIBC playground 2.0

Rated Challenge - echo (max 9 points)

Description
I made my own echo implementation, so that it's more efficient than the system one.
I noticed that sometimes it doesn't work as expected, but I'm sure it's just a minor bug.
Haven't ***GOT****ten the time to fix it!*

Points: *400*
Author: *carlo*

NOTE: To get the full **10** points, you also need to solve **echo 2.0**

Deadline: Tuesday, May 21st at 23:59

GL HF!

That's All
Folks