

Software Security 1

**Exercise Session**

24.10.2024

## Basics

quiz



| 32 solves

## Shellcoding

fizzbuzz



| 24 solves

peek



| 23 solves

peeky



| 22 solves

seashells



| 23 solves

deprecated



| 18 solves

# Quiz

```
~ $ nc 127.0.0.1 1024
```

```
Welcome to our quiz! You have 120 seconds to answer all questions!
```

```
Which of these is the architecture that considers code as being the same as data
```

- (1) Harvard architecture
- (2) von Neumann architecture
- (3) Le Corbusier architecture
- (4) Turing architecture

```
> 2
```

```
What is the name of the well-known debugger from the GNU project?
```

- (1) ida
- (2) windbg
- (3) lldb
- (4) gdb

```
> 4
```

# Quiz

- Spawn an instance on the scoreboard
- Access the task remotely with **nc** (netcat)

**Challenge:** Can you automate it?

# Fizz Buzz

## fizzbuzz



easy | 24 solves | 303 points

Shellcoding

In this task, you'll need to write some shellcode that will solve a variant of the famous FizzBuzz problem.

You'll receive a pointer to an array of 2048 unsigned 32-bit integers in `rdi`. Replace all values that are divisible only by 3 with zero, those that are only divisible by 5 with one, and those that are divisible by both with two. All remaining numbers should be replaced with three.

Then, return cleanly from your shellcode. If you correctly fizzbuzz-ed all the numbers, you'll receive the flag.

# Fizz Buzz

- ASM coding challenge
- **Tip:** break it down into smaller steps:
  - Write an ASM code that loops through an array
  - Write an ASM code that uses **div** and get the remainder
  - Debuggers like GDB (pwndbg) can be very helpful

# Peek

## peek



easy | 23 solves | 313 points

Shellcoding

Can you take a peek at the flag?

This is another simple shellcoding task. The shellcode runner has a built-in sandbox that will allow you to perform only the following system calls:

- open
- read
- write
- sendfile

Try to read the flag from `/flag`.

# Peek

- ASM coding challenge
- Use the **open** and **read** syscall to get the contents of **/flag**
- Use **write** or **sendfile** to output it to **stdout**



# Seashells

**seashells**



easy | 23 solves | 313 points

Shellcoding

Finally, actual shell code shellcode. The flag path is randomized, try spawning a shell!

# Seashells

- ASM coding challenge
- Use the **execve** syscall to run **/bin/sh**
- In the shell, use whatever (**cat**) to read **/flag**

# Peeky

**peeky**



easy | 22 solves | 323 points

Shellcoding


Can you do the same thing again? No null bytes allowed this time, though.

# Peeky

- Same as peek, but your shellcode can't contain \x00 (null character)
- Choose registers carefully
- Use idioms:
  - **push** and **pop** for “/flag” and other values
  - **xor <reg>, <reg>** to zero registers
  - **inc, dec, add**, and other operations to avoid null bytes
- **pwntools** `print(disasm(asm('your shellcode')))` can help

# Deprecated

## deprecated

 easy | 18 solves | 370 points  
Shellcoding

It's time for your first `_real_` exploit!

---

From `man 3 gets`: **Never use this function.**

# Deprecated

- It's a buffer overflow where you also have Address Space Layout Randomization (ASLR) enabled.
- **Tip:** you can disable ASLR locally to build shellcode incrementally, then find a way to jump to the correct address.
- Find the buffer overflow padding
  - You can use cyclic patterns within a debugger to help you
  - Use the leaked address to determine the stack position
  - Write a payload on the stack that overwrites the return address to jump to your shellcode (e.g., **execve /bin/sh**)
- **pwntools** can be really useful for creating cyclic patterns and inputting raw data into the application