Wargame 3 | COMP6447 20T2

The following wargames will provide you with exercises where you will be required to:

1. Write x86 assembly in the form of shellcode

2. Reverse engineer an IDA flow graph into C code

You can download the challenges here: https://cloudstor.aarnet.edu.au/plus/s/lOtBqeEKEmjczI7

These challenges are a zip file with the password: DoYouEvenUseGent00?

There are **3 exploitation challenges** and **1 reverse engineering challenge** this week!

Try to solve the **exploitation** challenges locally first, then connect to our servers to obtain the flags. To get full marks you must get the flag from our servers.

For simple, and find-me, the goal is not to get a shell . Getting a shell will not get you full marks. The goal is to print the flag with shellcode.

It is expected that you write all shellcode for these challenges yourself, and submit the assembly version of the shellcode. Not just raw bytes

Challenge	IP:PORT
simple	plsdonthaq.me:3001
shellz	plsdonthaq.me:3002
find-me	plsdonthaq.me:3003

Each **exploitation** challenge has a flag to submit. The flag is in the format FLAG{XXX}. To get full marks in this wargame, you need to submit all flags.

Reverse engineering challenge

This week we also have a reverse engineering challenge. The challenge is in the form of a screenshot of binaryninja graph view output.

The screenshot is of a **single function** . You are required to fully reverse engineer this function into C code. You're submitted C code should contain a **completed** function, it shouldn't contain trivial errors such as typos, undefined variables. To get full marks for the reverse engineering challenge, your C code must follow the same logic as the C code we used to generate the image. It does not have to be exactly the same! **This RE challenge is similar to the RE challenge that will be in the final exam, so try to get it as close to the real C as possible.**

Variable and function names are up to your discretion. However if a function or variable name is obvious (can be found in the image) you should use these names. Any arguments should be shown for function calls

ie: the following two code segments are the same in our eyes.

```
int main() {
    int a = 10;
    printf("%d\n", a);
    return 1;
}
```

and

```
int main() {
    printf("%d\n", 10);
    return 1;
}
```

Submission Instructions

A markdown document (.md) containing the following for each challenge:

We are interested in proof that you understood the challenge, the vulnerabilities and how to exploit them. This is not intended as a formal bug report.

~~~
chal1
===========================
Flag: FLAG{hi}
General overview of problems faced
-------------------------------------
Had to hack the program
Script/Command used
------------------
```
print "hello_world"
```
re challenge
=============
General overview of problems faced
-------------------------------------
needed to use man page to find arguments for atoi
```C
int main(int argc, char** argv) {
    int a = atoi(argv[1][0]);
    if (a > 20) {
        printf("%d\n", a + b);
    }
}
```
~~~

Please submit the document as a markdown file on give. You may submit as many times as you like. Only your most recent submission will be marked.

Submission

```
give cs6447 war3 war3.md
```

Marking scheme

Each week's wargames are worth 3 marks in total.

Due date

The wargames are due **17:59 Tuesday 23rd June (Sydney time).** This is in Week 4

Late Penalty

Late submissions will have marks deducted from the maximum achievable mark at the rate of 1 mark *per day* that they are late.

Resource created about a month ago, last modified 5 days ago.