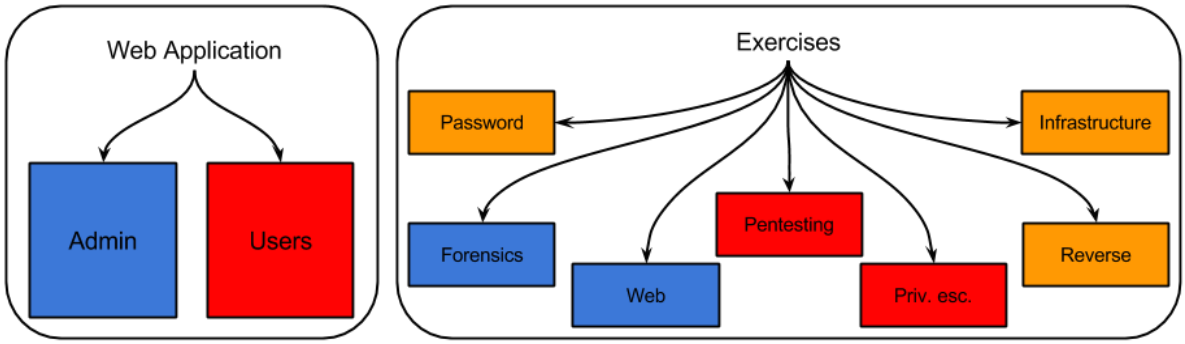
Individual report

by Steven Marien

1 - Introduction

This document will explain how we split the project and then how I contributed to it. Our project is easily splittable, as all parts do not interfere with each other. As a quick example to explain this: a virtual machine which emulates password attacks as nothing in common with basic forensics exercises.

Based on this assumption, the project looked like the following image.



Alexis (aggs3)
Steven(sm861)
Both of us

The web application that supports our exercises did not need to be finished for us to start our virtual machines, as it is designed only to share them to future students.

In the same idea, a virtual machine could easily be done by Alexis while I was doing another one at the very same time.

2 - Leader

Before we go further, it is important to note that we decided not to have a leader in our small group of 2. While having a leader can help attributing tasks and deciding whether or not a functionality is done, it did not seem fundamental on this project.
We have been working together with Alexis for a long time already and we knew that we were able to discuss ideas without the need of a leader.

3 - Exercises

Choosing the kind and nature of exercises was sometimes trivial and obvious, but sometimes a bit more experience was needed. For example, we both knew that the OWASP top 10 of web application vulnerabilities was an obvious choice.

Once we chose what could make good exercises, we split them accordingly to our skills and inclinations.

3 - 1 Privilege escalation

There are two machines that provide a privilege escalation vulnerability. I made the first one by installing a NFS service on the system, which I then configured in a way that authorises a user to write a file with root's rights. In order to build the second machine, I had to find an old and vulnerable version of Ubuntu running with a 2.6 kernel version, which, is known to be highly vulnerable.

3 - 2 Pen-testing

I also designed basics exercises that cover must-know facts in everyday pen-testing. They explain how to bind a shell from a victim computer to the attacker computer and the reverse equivalency. They therefore also introduce how NAT devices and firewalls can be by-passed if not configured well enough by providing an example of dangerous outgoing traffic.

3 - 3 Infrastructure pen-testing

These particular machines were really interesting to produce and the idea behind is, in my opinion, really important for beginners. Indeed, they are honeypots kind of machine, and expose the student with two different style of self-defence mechanism. I made them by installing the open source project Artillery on Debian Linux, which opens a few well known ports on the machine and triggers an alert if a TCP connection goes through. One machine is configured to ban any IP which looks suspicious while the other one just logs alert into a file, which is then presented to the student. They can therefore see that they were detected all along, and realise that real life systems will have powerful defence mechanisms as these.

3 - 4 Reverse engineering

We both worked on buffer overflow exercises in order to learn what we did not know about them. Once we knew enough, I made a small C program which execution flow can be altered. An overflow occurred on a user input which can be used to then redirect to a function, this function is the target and is not called normally.

4 - Web Application

For the web application that supports our project, we also decided to split the work. It is interesting to note that the web application is actually two web applications working together.
The student part, first of all, is a read only web-app that displays the exercises descriptions and allows to download the machines. The other one has been made for administrators and is available by designed only on the local host. I personally contributed as a front developper, for the read-only web-app, while Alexis was coding all the backend.

5 - Tests

Each of the machines we made have been produced and then tested by the person who made them. However, once we were done with them, the other one had to test them as well, as a user point of view.
This enforces that our machines were first of all possible from an external point of view and environment and that we both knew how to break them all and have enough knowledge about the subject, in order to both learn as much as possible from the it.

6 - Documentation

Fortunately for us, we both knew how to make videos as the ones provided in our turn-in folder. So we split these as well. The interesting thing here is that we actually made the video for the other one's machines, which was a good way to practice even more the machines we did not produce, as we made them after the testing part. We used the other files while we were working (for example, VM technical description) to help us organising our ideas and vulnerabilities, so it was a not a big deal to add them in the final corpus.