# System Security
# Graded Assignment 1 - Exploits

**Graded Assignment**

Distribution: 29.09.2022
Submission deadline: 13.10.2022 h14:00

## 1 Introduction

The goal of this exercise is to understand and perform a simple multi-step exploit that leverages a set of known vulnerabilities and misconfigurations. We will see how some common mistakes (either programming errors, or security misconfigurations) can be exploited, both manually and through automated tools.

The idea is to get a flavor of why patching, proper configuration, and secure coding are important: most security issues arise from the same well-known set of bugs, which will be the first tested issues by attackers. The same holds for updates to exposed software: once more complex, *0-day* vulnerabilities get disclosed, they are quickly available (and easy to replicate) by anyone.

### 1.1 Goals and Setup

This exercise will exploit a website and its server, from bugs in the server to complete escalation of privilege and access to the server. Your ultimate goal is to compromise the server by stealing the SSL certificate in use, located in `/usr/lib/ssl/private/secret.crt`. As we cannot expose such a vulnerable service on the Internet, we will use `Docker` to host the webserver as an isolated container on your VM.

You can use the VM to run the exercises:

- Download the exercise zip archive from Moodle and extract it in the VM.
- Navigate to the folder with a terminal; build the exercise with the command `sudo ./build.sh`. Be patient: the VM has to update itself, and the Ubuntu version inside the container has to update as well.
- Logout and login again, or reboot the VM.
- To check the IP address of your server, you can issue `docker inspect exploits | grep IPAddress` and note down the output.

If you have Docker on your own system, you can avoid using the VM, and simply build and run the container on your own machine; however, note that we cannot offer technical support in this case if the exercise does not work as expected.

The server, hosting a website that you can access from the machine running Docker, can be found at the address `https://IP_ADDRESS:443`.

**Troubleshooting:**

- It is important to use the correct scheme, i.e., `HTTPs`, and to write it down explicitly in the browser URL bar.
- You will have to accept the self-signed certificate from the browser.
- Be patient if the VM just booted: it takes a few seconds for the server to (re)-start.
- If the server does not seem to be online (i.e., *Unable to connect*), type `docker ps` in any terminal.
- Do not update the VM! Please start from the state of the VM when you download it from the course Moodle.
- If you see "Secure Connection Failed - `PR_END_OF_FILE_ERROR`" when accessing the website, please issue `docker restart exploits` on a terminal.

## 2 Manual Exploit

1. The OWASP Top 10 "represents a broad consensus about the most critical security risks to web applications". It is a very popular list of vulnerable patterns for web security. Consult the 2017 list here: `https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf`.

   Check the source code of our vulnerable website. Do you see any potential vulnerability that allows us to read any arbitrary file on the remote filesystem? For example, a value controlled by the user that is used without checks from a function that reads files. What OWASP Top 10 category would you classify it as?

2. Try to read some filesystem files that you are not supposed to. How do you get them exfiltrated thanks to the previous vulnerability?

   Let's get some interesting files. How many users are there in the system? How can you tell by just being able to read files? How many users can log in into the system?

3. Let's try to get the target file. Does it work? Why not? Can we discover who we are running as? (*Hint: some files in `/proc/self` might help you – consult the `proc` documentation by checking its manpages*).

4. There is a big misconfiguration in the server that allows us to get the (hashed) Unix user passwords. What is this misconfiguration? How many users have a password?

   Let's crack the user password: use the tool *John The Ripper* to crack it. To use John, you need a *dictionary* of passwords that it can try to hash and compare to the target hash. You can start with an English dictionary of words, for example `https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt`, otherwise, you can use popular password leaks such as *rockyou.txt*, or other dictionaries. What is the user's password?

5. Let's see if we can use our new knowledge: use `nmap` to see how many open ports the server has. What command do you have to issue? What is the other exposed service, besides the webserver?

6. Login as the user in the system. Can you get the target file? What groups are you part of – is there any interesting group? You should be able to get the target file – and control the full system! What is the SSL certificate of the server?

# 3 Metasploit

We will now get to the same result using a shorter route. Instead of manual exploitation and code auditing, we will observe how once a big vulnerability is public domain, then anybody can exploit it. Hopefully, you will see why timely patching of software is important.

We will use a popular penetration testing framework: *Metasploit*. Metasploit provides pre-made modules that allow testing and exploitation of popular vulnerabilities and misconfigurations. It is already installed in the VM, and its CLI can be accessed by typing `msfconsole` in a terminal.

You can learn how to use the metasploit framework at `http://www.metasploit.com/`. Here are a few useful commands:

- "search application-name" will give you all possible exploits available for a given application. Try and understand which of them best suit your requirements. There is plenty of help out there on the Internet!

- "show actions" lists the possible actions for the loaded module, while "show options" lists the information required for the exploit (e.g., target machine, port, etc.). You can then set actions and options with the `set` command.

Our target server is running a very old and unpatched Ubuntu GNU/Linux distribution from 2013. In particular, the OpenSSL version is `OpenSSL 1.0.1c 10 May 2012`. Check the known vulnerabilities of this version of OpenSSL, for example on the Common Vulnerabilities and Exploits (CVE) database. The CVE database provides a method of tracking all publicly known exploits and security issues of software, down to every version and release of software. Is there a very famous vulnerability known for this version of OpenSSL? Find the Metasploit module that can exploit it, and use it to get the private key. Is it the same key we leaked with the previous exploit? If not, why?