**Alma Mater Studiorum**

***Universita di Bologna***

**ETSETB**

***Universitat Politecnica de Catalunya***

# Cybersecurity threats and mitigation in Web Applications

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Bachelor of Telecommunication Sciences**

by

**Anna Solina Perez**

to the Faculty of Informatics at the Alma Mater Studiorum

Advisor: Dr. Franco Callegati, UNIBO
Dr. Oscar Esparza Martin, UPC
Assistance: Dr. Andrea Melis, UNIBO

June 2022

# Abstract

Every day IoT is being used more and more, they connect from our fridge to our security cameras. Recent studies have shown that they is a huge security hole, especially with the web interfaces. Web interfaces are really difficult to secure, as there aren't still any specific guidelines standard to completely secure them.

In this thesis we can find a theoritical part where we can see the methods used in Cybersecurity and we complement it with a practical part to be able to put into practice everthing we have learnt.

# Resum

Cada dia s'utilitza més el IoT, que connecta des de la nostra nevera fins a les nostres càmeres de seguretat. Estudis recents han demostrat que són un enorme forat de seguretat, especialment les interfícies web. Les interfícies web són realment difícils d'assegurar, ja que encara no hi ha directius específiques standards assegurar-les completament.

En aquesta tesis podem trobar una part teòrica on podem veure els mètodes utilitzats en Ciberseguretat i la complementem amb una part pràctica per a poder posar mostrar el funcionament de tot l'après.

# Resumen

Cada día se utiliza más el IoT, que conecta desde nuestra nevera hasta nuestras cámaras de seguridad. Estudios recientes han demostrado que son un enorme agujero de seguridad, especialmente las interfaces web. Las interfaces web son realmente difíciles de asegurar, ya que todavía no hay directrices específicas standard para asegurarlas de forma completa.

En esta tesis podemos encontrar una parte teórica donde podemos ver los métodos utilizados en Ciberseguridad y la complementamos con una parte práctica donde podemos ver el funcionamento de todo lo aprendido.

# Acknowledgements

First of all, I want to thank to my supervisors Franco Callegati and Andrea Melis, who have been there for me and helped me in every small problem or obstacle I have encountered. Also, to my supervisor of the UPC, Oscar Esparza to being close to me and giving me his best advice. To my roommates who have been there all the journey helping with motivation and encouragement and disconnection moments. To my parents, who have supported me for this semester in the fantastic University of Bologna and the beautiful city of Bologna.

# Revision history and approval record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 06/04/2022 | Document creation |
| 1 | 11/05/2022 | Document revision |
| 2 | 27/05/2022 | Document revision |
| | | |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Anna Solina | anna.solina@estudiantat.upc.edu |
| Franco Callegati | franco.callegati@unibo.it |
| Oscar Esparza | oscar.esparza@upc.edu |
| Andrea Melis | a.melis@unibo.it |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 10/06/2022 | Date | 17/06/2022 |
| Name | Anna Solina | Name | Franco Callegati |
| Position | Project Author | Position | Project Supervisor |

# Contents

# List of Figures

# List of Tables

# 1. Introduction

The IoT demand has been in a continued growth since its "beginning". This has lead the industry to make devices without proper consideration of their security. In 2017 it was estimated that 127 new IoT devices were connected to the internet every second [1]. As a result of the widespread use of insecure IoT, malicious attackers have an easy way into the internal network. In 2017 [2] hackers managed to extract 10GB of data from a North American casino. They managed to breach the casinos' network by first hacking into an internet-connected fish tank.

There are a lot of possible security holes in an IoT machine (or machines in general), but we will classify them in 3 main ways. Firstly, there are open services that are dangerous to used, for example telnet, specially if you forget to stop them when you are finish. Afterwards, there are also open services that are intended to be used but have vulnerabilities in them (for instance, because they are 0-days or because haven't been updated…). Last but not least there are the web servers, this opens up a plethora of security risks as coding a secure website is extremely difficult.

In this thesis we investigate the security of the websites in IoT devices. In order to create a framework that automatically audits the web pages of IoT devices, we need to emulate them.

Furthermore, in this thesis, we will be able to see how a web application can survive an attack and how it can be updated and its firmware upgraded. This is a very important need because as we have already said, all web applications with personal data are prone to receive these attacks.

# 2. Workplan

## 2.1. Incidences

My idea of doing this thesis was to learn more about the IoT world and how to make it more secure. I started in December by talking to Professor Callegati to discuss the topic.

Later, when I was already settled in Bologna, Italy, I started to meet him and other professors from the university.

We have done a part of theoretical training, a more practical part and in parallel, I have been documenting the whole thesis and complementing it with contests of pages that have been shared with me and with other knowledge that have helped me.

Now I am going to break down each work package to make it more understandable and I will also add the temporal data.

## 2.2. Update Work Packages

<table>
<tr><td colspan="2">Title: Formation in Web Application Vulnerabilities</td></tr>
<tr><td>Major constituent: Basic formation</td><td>WP ref: WP 1 of 3</td></tr>
<tr><td>Short description:<br><br>In order to discover and learn how it works the security of the IoT, first of all we have to know all the vulnerabilities to make possible to solve it.</td><td>Start event:<br><br>01/03/2022<br><br>End event:<br><br>15/04/2022</td></tr>
<tr><td colspan="2">Tasks:</td></tr>
</table>

| |
|---|
| - Work on the material send by the professor. [3] [4]<br>- Work on interactive pages that allow you to view vulnerabilities. [4] [5] [6]<br>- Complement all the material received with material that I find and/or doubts that I may have. [7] |

### 2.1 WP 1: Formation in Web Application Vulnerabilities

| Title: Working on the code of a Web Application | |
|---|---|
| Major constituent: Practical part | WP ref: WP 2 of 3 |
| Short description:<br><br>After having understood how it works and what its vulnerabilities are, to be able to see and make a code that corrects them or makes them more difficult for possible attacks. | Start event:<br><br>15/04/2022<br><br>End event:<br><br>30/05/2022 |
| Tasks:<br><br>- Understand the code send by the professor.<br>- Make my own code and solve new necessities. | |

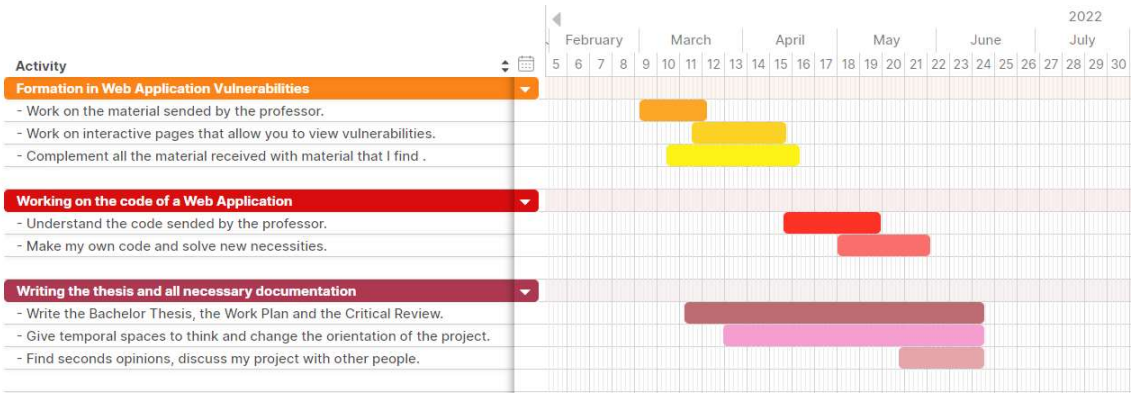
### 2.2 WP 2: Working on the code of a Web Application

| Title: Writing the thesis and all the necessary documentation | |
|---|---|
| Major constituent: Documentation | WP ref: WP 3 of 3 |
| Short description:<br><br>It may seem odd that there is this Work Plan but I would like to emphasize the importance of investing quality time in the writing of the documents, because it gives you moments of reflection to be able to analyse and change the focus of the thesis.<br><br>Also, it is a good moment for self-criticism and to check if you are fulfilling your objectives defined at the beginning of the project as well as the marked deadlines. | Start event:<br><br>15/03/2022<br><br>End event:<br><br>10/06/2022 |
| Tasks:<br><br>- Write the Bachelor Thesis, the Work Plan and the Critical Review.<br>- Give temporal spaces to think about it and correct/change the orientation of the project. | |

- Find seconds opinions, discuss my project with my super advisors and the classmates

*2.3 WP 3: Writting the thesis and all necessary documentation*

## 2.3. Updated Time Plan




*2.4 Updated Time Plan*

# 3. Background

Before starting, we are going to explain some of the main concepts necessary.

## 3.1. Internet of Things

The Internet of Things (IoT) is the network of daily objects connected to the Internet. The IoT concept was first proposed by Kevin Ashton in 1999 [8]. He talked about all of the objects (not humans) that create information for the internet. Today this idea has transformed the way we live and interact with technology, now we have a vast amount of computer generated information at the tip of our fingers. It's expected that by 2025 the amount of data generated by IoT devices will reach 73.1 ZB (where a 1 $1\,ZB = 2^{70} bytes$ ) [9].

## 3.2. Web Application Vulnerabilities

During the process of building and deploying your web applications, it has to be super important the data breaches. The cost of forgetting this it was $3.92 million in 2019, and many of these incidents could have got prevented with the right mindset and a comprehensive audit to ensure web application security vulnerabilities get addressed.

In order to look at each important vulnerability individually, we will follow the OWASP (Open Web Application Security Project) standard. [10]

For each vulnerability we will see what it is and how to solve it or make it less vulnerable. [7]

### 3.2.1. Injection Flaws

Injection flaws are when an attacker uses unfiltered and often malicious data to attack databases or directories connected to you web apps. Two common injection attacks often get used. First, SQL injection gets used to attack your databases. Second, LDAP injection gets used to attack directories.

Injection attacks use input fields that interact with directories and databases to execute against vulnerabilities. These include usernames, passwords, and other areas that interact with the target. These fields are often left vulnerable due to the lack of an input filter when the database or directory's development.

Protecting your web application from injection attacks can be a simple fix. Adding filters to your inputs is the best defense. With SQL databases, we can first use prepared statements that can help prevent attackers from manipulating queries. Second, with LDAP injections, we can use protocols like escape variables to prevent characters used with injection attacks from being passed to manipulate the directory.

### 3.2.2. Broken Authentication

The process of authentication helps apps identify and validate users. Therefore, broken authentication can allow attackers to access and have the same permissions as the targeted user, creating severe web app vulnerabilities. Issues with authentication can give an attacker unfettered access to your data and wreak havoc on your web application.

Authentication vulnerabilities can include improperly hashed and salted passwords, leaks involving user account data, improperly set timeouts, brute force attacks, or typical password stuffing like password1 or admin1234.

There are ways we can help to prevent authentication vulnerabilities. Using multi-factor authentication can help verify the correct user. Creating strong passwords with periodic password updates can keep from common password use. Finally, properly configuring timeouts and password security within your database will prevent authentication issues.

### 3.2.3. Sensitive Data Exposure

Sensitive data gets transported or stored without any encryption or other protection, leaving information vulnerable to various attacks.

When we talk about unprotected data, it can be understood in two ways. First, while data is transported from the user to the client, attacks as a man-in-the-middle attack can be used to steal data from packets. Second, stored data, while more complicated, can be exposed through encryption keys get stored with data or weak salt/hash or passwords and credentials.

To prevent the exposure of your sensitive data is vital to the security of your app. Due to data vulnerabilities in motion, HTTPS, and perfect forward secrecy (PFS), ciphers need to get implemented for incoming data to your site. Disabling data caching that may store sensitive information is another way to help protect data.

In addition to transported data, stored data is at risk for attacks and exposure as well. Encrypting data stored in your databases while keeping encryption keys stored separately will reduce exposure. Eliminating out-dated data or data that isn't needed will minimize exposure. If there is no data, there is no risk.

### 3.2.4. Missing Function Level Access Control

When server-side authorization is misconfigured, broken, or missing vulnerabilities will occur that can leave your back-end open to attacks.

These attacks often happen with front-end UIs configured with components to give admins access to data or other vital app elements. In this case, most users can't see the admin function, but those looking to find vulnerabilities will be able to uncover and exploit this flaw with malicious requests.

Fixing this flaw is simple. All server-side authentication needs to be active and configured to prevent unwanted access.

### 3.2.5. Security Misconfiguration

In the actually, often web applications are misconfigured, leaving an array of vulnerabilities for attackers to capitalize. Security misconfigured vulnerabilities can include unpatched flaws, unused pages, unprotected files or directories, outdated software, and running software in debug mode.

All aspects of your web applications can be affected by security misconfigurations. When a misconfiguration is found, it is vital to run a security audit to check for attacks or breaches.

There are ways we can help to prevent security configuration vulnerabilities. For instance, using a deployment protocol to continuously develop and deploy updates inside a secure environment or segmented application architecture will help prevent security vulnerabilities. Automatic your deployment will also keep your applications up to date and prevent attacks.

### 3.2.6. Cross-Site Scripting XSS

Cross-site scripting uses malicious code injected into benign sites to attack a user's web browser. An attacker will insert the code through a link and, together with social engineering, will lure the user to clicking the link and executing the code. Attackers using JavaScript for XSS vulnerabilities can access a user's webcam, location, and other sensitive data and functions.

XSS vulnerabilities are common where input is unsanitized. Additionally, XSS can allow attackers to steal cookies from users' browsers and access browsing history and sensitive information.

Ultimately XSS vulnerabilities can be fixed by sanitizing input. Sanitizing input will help stop user input from manipulating vulnerabilities and injecting them into websites. Also, validating and escaping user input will help prevent malicious injection.

### 3.2.7. Insecure Direct Object References

When database keys or files get exposed to the user, insecure direct object reference vulnerabilities exist. Because of the exposed internal objects, attackers can use enumeration attacks to access those objects and gain data or access to sensitive databases. Often authentication is either non-existent or broken.

Database objects are often vulnerable through URL parameters exposing serialized data keys an attacker can manipulate to access information. Also, static files can be manipulated and changed by an attacker to access sensitive information or other user's data.

Preventing access to sensitive files and databases can be done with server-side input validation. Testing input server-side can help prevent malicious user's from manipulating URL parameters and file names. Also, access control

measures can help determine if the user has permission and can access or change files and databases.

### 3.2.8. Cross-Site Request Forgery

Cross-site request forgeries (CSRF) use social engineering to trick authenticated users into clicking a link, as an example and take control of their sessions. Due to having authenticated sessions, the attacker can perform changes to the state of an app vs. data theft.

Applications without the proper dual authentication or cross-site tokens can be vulnerable to CSRF attacks. Those will little knowledge of social engineering are also at higher risk of their authenticated sessions hijacked.

There are several preventative measures to help stop CSRF attacks. Using secret tokens or cookies can help with authenticating real requests vs. malicious ones. Also, utilizing POST requests only and eliminating GET requests can help keep the URL information from getting compromised.

### 3.2.9. Using Components with Known Vulnerabilities

Due diligence needs to get done when considering using a third-party code or component in your web application. Many security issues can come with using unfettered code from sources you aren't familiar with.

To help find what components may be vulnerable, the National Vulnerability Database has a comprehensive list of known third-party vulnerabilities to help make the best choice.

Every aspect of your app can be affected by vulnerabilities in third-party code. For example, backdoors can get added to financial services code allowing attackers access to sensitive data.

The best way to prevent using vulnerable code is to know where and who it's coming from.

### 3.2.10. Unvalidated Redirects & Forwards

Unvalidated redirects and forwards is another input manipulation vulnerability again using parameters like GET requests to execute the attacks.

An example of the vulnerability is an attacker manipulating a URL and redirecting users to a malicious site where information can get stolen using social engineering and links with malicious code or links.

By eliminating redirects, you can eliminate the issue of redirect attacks. If necessary, keep redirects and forwards static, not allowing users to input URLs.

# 4. Implementation

As I already mentioned, this thesis is written to obtain more knowledge about the vulnerabilities of the Internet of Things and to be able to solve them or make them more robust.

We are going to work from diferents Challenges that are each of them focused on a different vulnerability. We will be guided by a Githlab repository of professor Andrea Melis from the University of Bologna [11]. In this repository we have some Python Scripts that, as we have already mentioned, are challenges to get hidden information from a web page.

All the changes made are in my Github repository or in the Annex. There is my whole project and all the modifications of the original one. [12]

The methodology we will follow will be to work with a Kali Linux environment via virtual machine and with the Python tool. [13] [14]

## 4.1. Procedure

The procedure to be followed will be the following.

There will be a series of challenges and in each one, there will be a part of written resolution and also a graphic that also explains the resolution of the challenge. That is to say, we will have the written part with more explanations and easier to understand, and then we will complement it with a graphic that will be much more visual and we will be able to see the general process better.

As we have already said, all the code is in the repository and we can log in to get it and try to run it by ourselves or get it from the Annex.

Each challenge is oriented to work on a specific vulnerability, so we will try to work on as many vulnerabilities as possible. Thus, our aim is to familiarize

ourselves with the challenges and make them our own by modifying some aspect.

## 4.2. Analysis and Resolution

In the repository, there are three challenges. In this section, we will explain them and solve them in order to learn how it works and improve them.

**web_Command_Execution_Challenge**

This challenge refers to the vulnerabilities described at the point Injection Flaws.

The challenge is to exploit a command injection.

Through a GET request to the root / with the parameter domain, it is possible to make a dns request to a site, e.g. `/?domain=www.ulisse.unibo.it`.

The goal of the exercise is to exploit the command injection and retrieve/read the first ten lines and the last ten lines of the /etc/passwd file.

The first step is to see that if you run:

http://localhost:8000/?domain=www.ulisse.unibo.it

You will see that in the browser you can see that the server is not able to give you the information correctly. However, if you run `dig www.ulisse.unibo.it` in the terminal, you can see the DNS information.

The next step is to check that you can run two commands at the same time, but of course, you have to take into account that the browser has not been able to deliver the correct DNS information, so if you run `http://localhost:8000/?domain=www.ulisse.unibo.it && ls` in the browser it will not work because it cannot do both commands. Instead, we have to use "`|`" as it will do only one.

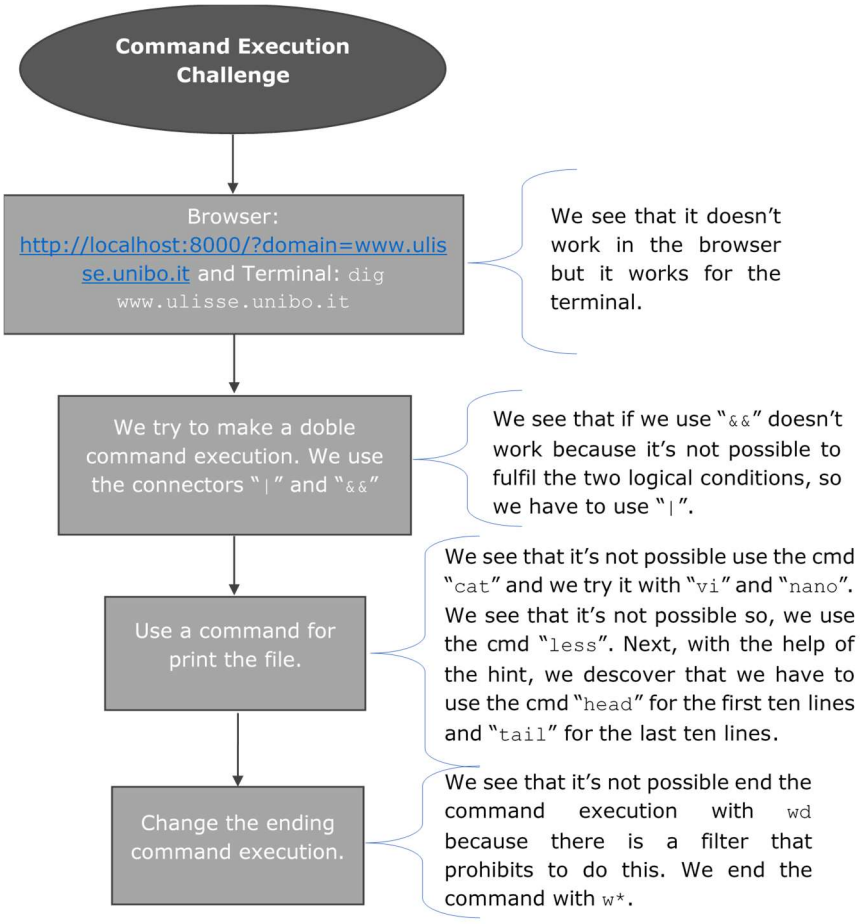
Next, you can see that there are quite a few filters to print the file in question. For example, if you use the `cat` command, you will get a response specifying that the command used is forbidden.

We are looking for commands that allow us to view the file. If we use commands that are used to edit the files like `vi` or `nano`, we see that they are also disabled. Although it seems that this could be a viable solution, as file editors they do not print as requested.

Another option we have is to use the command `less`, we see that this is also filtered but we get a hint: it makes us think that we only have to print the first and last ten lines of the file. After that, we can think of using the commands `head` and `tail`. With these two commands we can get the correct answer.

Finally, we see that the challenge has a filter that prohibits us from seeing files that end in wd, so the solution is to use `head /etc/passw*` and `tail /etc/passw*`.

Therefore, the answer to the exercise is to run the following line in the explorer: `http://localhost/?domain=www.ulisse.unibo.it | head /etc/passw*` and `http://localhost/?domain=www.ulisse.unibo.it | tail /etc/passw*`.




*4.1 Command Execution Challenge*

**Web_SQLi_Challenge**

This challenge refers to the vulnerabilities described at the point Injection Flaws and Broken Authentication

The challenge is to exploit a SQL injection to bypass the login.

The login is performed by a GET call to the address /login with the parameter's username and password, for example `/login?username=&password=`.

The goal of the exercise is to log in as the administrator user.

For the exercise to be successful, the student must deliver: the FLAG, the complete query that exploits the SQLi, a brief description of the steps performed, why the vulnerability exists and how it is exploited.

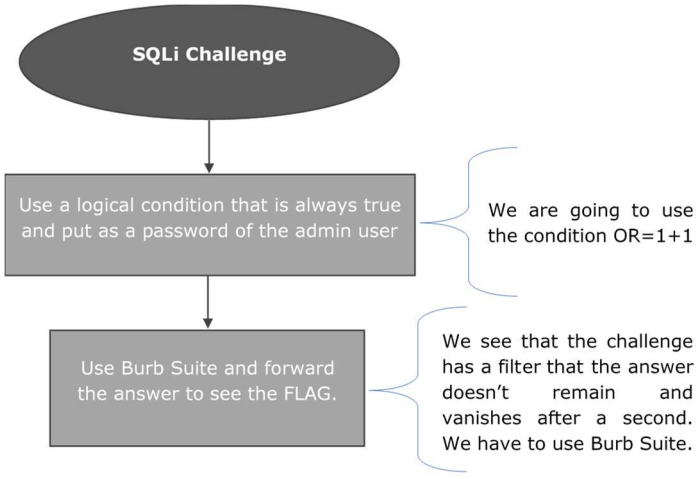
For this challenge you need to be trained in SQL injection and how to circumvent it. Thanks to the documentation obtained by the teachers, the challenge has been solved [15].

The secret is to understand that the injection must always be correct. This is solved by adding a condition that is always true. For example, `OR 1=1`.

If we add this in the search engine and mark that the user is the admin we will get the answer. So, the line we have to put in the search engine is the following: `localhost:8000/login?username=admin&password='OR'1'='1`.

It is important to understand how the password condition works. That is, understand how the apostrophes work and what they are commenting out.

Finally, when inserting the answer in the browser, we can't see the flag. This is because it is activated so that the answer does not remain and vanishes after a second. This obstacle can be easily solved, one of the many options, is to use Burb Suite and forward the browser to obtain the static answer and be able to see the answer [16].




*4.2 SQLi Challenge*

**Web_Path_Traversal_Challenge**

This challenge refers to the vulnerabilities described at the point Sensitive Data Exposure

The challenge is to exploit a LFI (Local File Inclusion)

Through a GET request to the root / with the path parameter it is possible to open a local file e.g. `/?path=file_locale.txt`.

The purpose of this exercise is to exploit the LFI and retrieve/read the file /etc/passwd.

There are some filters on the characters that you can send as a path.

This challenge aims to provide the user with knowledge of file paths in Ubuntu and how to move through directories using the terminal.

As the statement says, the challenge has quite a few filters and these will help you to solve it.

First of all, we have to try the obvious answer, to put the file directly. That is, use `http://localhost/?path=/etc/passwd`. In this case we see that we don't get the answer because there is a filter.

Then, we have to think that if we are in a directory mobility exercise we must use the following tool: `../` or `/..` since they are very used techniques to make the jumps in the directories.

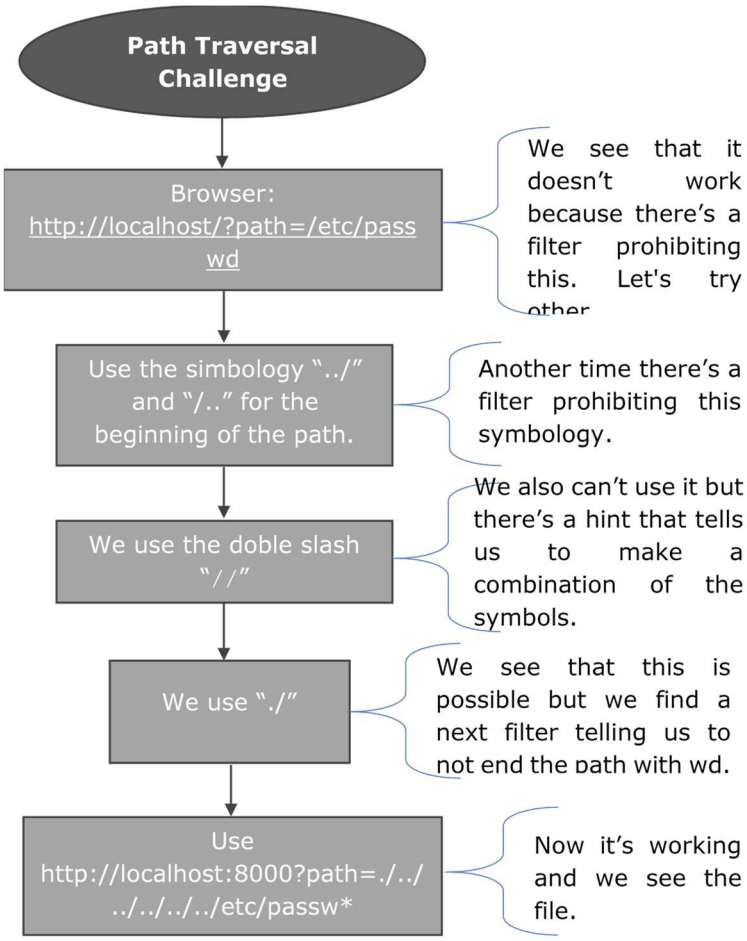
Even so, when we use them we see that it is not possible because there is a filter that forbids it. Then we can try that with the use of the double slash it will work (`//`), but we get another filter that denies us the answer. This filter tells us that we could try the combination of the two signs.

So, we try `./` and see that there is a new filter telling us that we can't end the line with the `wd` parameters. So, we modify the line with the asterisk `*` parameter.

Then, we see that it does work, that is to say, that no filter comes out, but it is still not the answer because the file has to be printed on the screen, and what we see is a few lines of code that can indicate that we are on the right track, but it is not the definitive answer.

At this point, we have to think as if we were in the terminal and start making jumps in the directories, that is to say, use the ../ tool to go jumping directories. If you don't know how many jumps you have to make, you can add them consecutively until you get the answer or understand where the folder etc. is located and know that there are five jumps.

This means that the URL that you have to put in the search engine is `http://localhost:8000/?path=./../../../../../etc/passw*`.




*4.3 Path Traversal Challenge*

# 5. Budget

During the thesis the tools used have been a HP laptop, which could have been any computer with no limitation in the needed specifications, and all of the software used is free and open-source like Docker and Python.

Then the main costs of the thesis come from the hours spent working on it by the student (me) and the supervisors. Taking into consideration the wages in Italy, 25 hours/week for the student and 2 hours/week for the supervisors and the fact that the project had a duration of 15 weeks combined we get the calculations from table 5.1

| Position | Amount | Wage/hour | Dedication |
|---|---|---|---|
| Junior engineer | 12 €/h | 25 h/week | 4500 € |
| Senior engineer | 45 €/h | 2 h/week | 1350 € |
| Total: | | | 5850 € |

*5.1 Calculations for the budget*

# 6. Conclusion and future development

Internet security is very extensive and it is very complicated to work in a secure environment. It is very important that these studies exist as they try to make the workspace more secure. Internet of Things is very powerful as it offers many possibilities but at the same time we have to be aware of what we upload and how we protect the uploaded data.

Summarising the project, what we can see is the implementation of challenges in order to learn by solving them. Thanks to these challenges, we can understand how vulnerabilities work and we can strengthen them by making them less vulnerable. We have seen that some challenges work through filters and others work through a loophole in the description. What I want to show in this paper how important is to perfectly know how the attacks work, in order to better protect ourselves from them. A good way to do this is the one shown in this paper. That is to say, in this work we have made the challenge more difficult by putting more filters but leaving a possible solution, the objective of cybersecurity is that there is no possible solution and that nobody can access the data, except those who have the permissions to do so, of course.

There are many recommendations in this sector, awareness campaigns for workers, more secure and renewable passwords, activate a URL filtering system, put double authentication for users... All these actions favour a more secure space but the most important is to make it as difficult as possible for a possible hacker to enter. In my opinion, the most effective is what we have done in this project. Trying to hack the web interface and see its weaknesses. This way, we can get to know much better what is efficient and what it is not so efficient.

For example, a good way to put more security into your web application is authentication. This is the first step in access control, and there are three common factors used for authentication: something you know (such as a password), something you have (such as a smart card), something you are (such as a fingerprint or other biometric method). This article provides you

with good understanding of the three factors of authentication and how they can be used together with multifactor authentication [17].

You should have a good understanding of the three factors of authentication (something you know, something you have, and something you are). You should also understand how they can be used together with multifactor authentication.

# 7. Bibliography

[1] M. Patel, «What's new with the internet of things?,» 2017. [En línia]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things.

[2] A. Schiffer, "How a fish tank helped hack a casino," 2017. [Online]. Available: https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino/?noredirect=on.

[3] D. Stuttard i M. Pinto, The Web Application Hacker's Handbook.

[4] «PortSwigger,» [En línia]. Available: https://portswigger.net/web-security/all-materials.

[5] «Over the Wire,» [En línia]. Available: https://overthewire.org/wargames/natas/.

[6] «Try Hack Me,» [En línia]. Available: https://tryhackme.com/.

[7] A. Dziuba, «10 Common Web Application Security Vulnerabilities and How to Prevent Them,» [En línia]. Available: https://relevant.software/blog/web-application-security-vulnerabilities/.

[8] K. Ashton, «That 'internet of things' thing,» 2009. [En línia]. Available: https://www.rfidjournal.com/that-internet-of-thingd-ting.

[9] B. Jovanovic, «Dataprot,» 2022. [En línia]. Available: https://dataprot.net/statistics/iot-statistics/.

[10] «OWASP Fundation,» [En línia]. Available: https://owasp.org/.

[11] A. Melis, «Githlab of Web Challenges,» [En línia]. Available: https://gitlab.com/wild_boar/labsec_course/-/tree/master/web_challenges.

[12] A. Solina, «Github,» [En línia]. Available: https://github.com/annasolina/Bachelor-Thesis.

[13] KALI, «Site to download the Kali Linux system,» [En línia]. Available: https://www.kali.org/get-kali/.

[14] Python, «Site to download Python,» [En línia]. Available: https://www.python.org/downloads/.

[15] A. Melis, «SQL injection,» [En línia]. Available: https://gitlab.com/ULISSecurity/teaching-material/-/blob/master/security-web/04_sql_injection/04_slide_web_sec2_melis_sqli.pdf.

[16] Portswigger, «Burp,» [En línia]. Available: https://portswigger.net/burp.

[17] «Understanding the Three Factors of Authentication,» [En línia]. Available: https://www.pearsonitcertification.com/articles/article.aspx?p=1718488.

# 8. Annex

In this section, we find the Python project carried out for this thesis. Below we can see the different codes and we can complement them with the explanation of point Implementation to understand them better.

**Web_Command_Execution_Challenge.py**

```
#!/usr/bin/env python
import BaseHTTPServer, cgi, cStringIO, glob, httplib, json, os, pickle, random, re, SocketServer, sqlite3, string, sys, subprocess, time, traceback, urllib, xml.etree.ElementTree
try:
    import lxml.etree
except ImportError:
    print "[!] Successful execution connected to http:/localhost:8000 \n If you want to run it locally install 'python-lxml' (e.g. '%s')\n " %("apt install-lxml" if not subprocess.mswindows else "https://pypi.python.org/pypi/lxml")

NAME, VERSION, GITHUB, AUTHOR, LICENSE = "Web Challenge Examination 10 September 2021 < Command Execution Challenge", "0.01", "https://gitlab.com/wild_boar/labsec_course", "@wild_boar", "GPLv3"
LISTEN_ADDRESS, LISTEN_PORT = "0.0.0.0", 8000
HTML_PREFIX, HTML_POSTFIX = "<!DOCTYPE html>\n<html>\n<head>\n<style>a {font-weight: bold; text-decoration: none; visited: blue; color: blue;} ul {display: inline-block;} .disabled {text-decoration: line-through; color: gray} .disabled a {visited: gray; color: gray; pointer-events: none; cursor: default} table {border-collapse: collapse; margin: 12px; border: 2px solid black} th, td {border: 1px solid black; padding: 3px} span {font-size: larger; font-weight: bold}</style>\n<title>%s</title>\n</head>\n<body style='font: 12px monospace'>\n<script>function process(data) {alert(\"Surname(s) from JSON results: \" + Object.keys(data).map(function(k) {return data[k]}));}; var index=document.location.hash.indexOf('lang='); if (index != -1) document.write('<div style=\"position: absolute; top: 5px; right:
```

```
5px;\">Chosen language: <b>' + decodeURIComponent(document.location.hash.substring(index + 5)) + '</b></div>');</script>\n" % cgi.escape(NAME), "<div style=\"position: fixed; bottom: 5px; text-align: center; width: 100%%;\">Powered by <a href=\"%s\" style=\"font-weight: bold; text-decoration: none; visited: blue; color: blue\" target=\"_blank\">%s</a> (v<b>%s</b>)</div>\n</body>\n</html>" % (GITHUB, "@wild_boar", VERSION)
USERS_XML = """<?xml version="1.0" encoding="utf-8"?><users><user id="0"><username>admin</username><name>admin</name><surname>admin</surname><password>7en8aiDoh!</password></user></users>"""
CASES = (("Blind SQL Injection (<i>boolean</i>)", "?id=2", "/?id=2%20AND%20SUBSTR((SELECT%20password%20FROM%20users%20WHERE%20name%3D%27admin%27)%2C1%2C1)%3D%277%27\" onclick=\"alert('checking if the first character for admin\\'s password is digit \\'7\\' (true in case of same result(s) as for \\'vulnerable\\')')", "https://owasp.org/www-community/attacks/Blind_SQL_Injection"), ("Blind SQL Injection (<i>time</i>)", "?id=2", "/?id=(SELECT%20(CASE%20WHEN%20(SUBSTR((SELECT%20password%20FROM%20users%20WHERE%20name%3D%27admin%27)%2C2%2C1)%3D%27e%27)%20THEN%20(LIKE(%27ABCDEFG%27%2CUPPER(HEX(RANDOMBLOB(300000000)))))%20ELSE%200%20END))\" onclick=\"alert('checking if the second character for admin\\'s password is letter \\'e\\' (true in case of delayed response)')", "https://owasp.org/www-community/attacks/Blind_SQL_Injection"), ("UNION SQL Injection", "?id=2", "/?id=2%20UNION%20ALL%20SELECT%20NULL%2C%20NULL%2C%20NULL%2C%20(SELECT%20id%7C%7C%27%2C%27%7C%7Cusername%7C%7C%27%2C%27%7C%7Cpassword%20FROM%20users%20WHERE%20username%3D%27admin%27)", "https://owasp.org/www-chapter-belgium/assets/2010/2010-06-16/Advanced_SQL_InjectionV2.pdf"), ("Login Bypass", "/login?username=&amp;password=", "/login?username=admin&amp;password=%27%20OR%20%271%27%20LIKE%20%271", "https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF"),("Server Side Request Forgery", "/?path=", "/?path=http%3A%2F%2F127.0.0.1%3A631" if not subprocess.mswindows else "/?path=%5C%5C127.0.0.1%5CC%24%5CWindows%5Cwin.ini", "http://www.bishopfox.com/blog/2015/04/vulnerable-by-design-understanding-server-side-request-forgery/"), ("Cross Site Request Forgery", "/?comment=", "/?v=%3Cimg%20src%3D%22%2F%3Fcomment%3D%253Cdiv%2520style%253D%2522color%253Ared%253B%2520font-weight%253A%2520bold%2522%253EI%2520quit%2520the%2520job%253C%252Fdiv%253E%22%3E\" onclick=\"alert('please visit \\'vulnerable\\' page to see what this click has caused')", "https://owasp.org/www-community/attacks/csrf"), ("Arbitrary Code Execution", "/?domain=www.google.com", "/?domain=www.google.com%3B%20ifconfig" if not subprocess.mswindows else "/?domain=www.google.com%26%20ipconfig", "https://en.wikipedia.org/wiki/Arbitrary_code_execution"), ("Path Traversal", "/?path=",
```

```
"/?path=..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd" if not
subprocess.mswindows else
"/?path=..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Cwin.ini",
"https://www.owasp.org/index.php/Path_Traversal"))

def init():
    global connection
        connection = sqlite3.connect(":memory:", isolation_level=None,
check_same_thread=False)
    cursor = connection.cursor()
        cursor.execute("CREATE TABLE users(id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT, name TEXT, surname TEXT, password
TEXT)")
    cursor.executemany("INSERT INTO users(id, username, name, surname,
password) VALUES(NULL, ?, ?, ?, ?)", ((_.findtext("username"),
_.findtext("name"), _.findtext("surname"), _.findtext("password")) for _ in
xml.etree.ElementTree.fromstring(USERS_XML).findall("user")))
     cursor.execute("CREATE TABLE comments(id INTEGER PRIMARY KEY
AUTOINCREMENT, comment TEXT, time TEXT)")

class ReqHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_GET(self):
        path, query = self.path.split('?', 1) if '?' in self.path else (self.path, "")
            code, content, params, cursor = httplib.OK, HTML_PREFIX,
dict((match.group("parameter"),
urllib.unquote(','.join(re.findall(r"(?:\A|[?&])%s=([^&]+)"
match.group("parameter"), query))))
for match in
re.finditer(r"((\A|[?&])(?P<parameter>[\w\[\]]+)=)([^&]+)", query)),
connection.cursor()
        try:
            if path == '/':
                if "domain" in params:
                    paths=str(params["domain"])
                    if re.search(";",paths):
                        content = "We filtered out the ';' character .. try harder!"
                    elif re.search("cat",paths):
                      content = "We disabled the 'cat' command... other commands
exist!"
                    elif re.search("wd$",paths):
                       content = "Error! The path cannot terminate for \"wd\", what
can you use to ovreride the +wild+ filter?"
                    elif re.search("nano",paths):
              content = "We have also disabled the 'nano' command!"
                    elif re.search("vi", paths):
                       content = "sorry, you cannot use this command... but this is
a command to edit files, are you sure you need this?"
                    elif re.search("less",paths):
                        content = "We have disabled the 'less' command! But, are
you sure you need it if you only have to read the first 10 lines and the last
10?"
                    else:
                        # print glob.glob(paths)
```

```
                    # final = glob.glob(paths)[0]
                    # print final
                        content = subprocess.check_output("nslookup " +
params["domain"], shell=True, stderr=subprocess.STDOUT,
stdin=subprocess.PIPE)
                elif "redir" in params:
                    content = content.replace("<head>", "<head><meta http-
equiv=\"refresh\" content=\"0; url=%s\"/>" % params["redir"])
                if HTML_PREFIX in content and HTML_POSTFIX not in content:
                    content +="<div><span>Web Challenge Examination 10
September 2021:</span></div>\n"
                content +="<p> The Challenge consists of exploiting a command
injection </p>"
                content +="<p> Through a GET request to the root / with the
domain parameter, it is possible to make a dns request to a site. e.g.
/?domain=www.ulisse.unibo.it</p>"
                    content +="<p> The purpose of the exercise is to exploit
command injection and retrieve/read the first 10 lines and the last 10 lines
of the file /etc/passwd</p>"
                content +="<p> There are some character filters that you can
use </p>"
                content +="<p> For each \"matched\" filter, a small hint will
be given </p>"
                content +="<p> The student must then hand over 3 files </p>"
                content +="<p> 1) The file report.txt where he/she will have
to explain in a understandable way the concept behind the vulnerability of
this challenge.</p>"
                content +="<p> The student should also list the various steps
and attempts made to exploit the vulnerability, with the reasoning behind
\"bypassing\" the filters, including of course the final payload. </p>"
                content +="<p> 2) The screenshot where you can clearly see
the call to the application with the payload and the final result. </p>"
                content +="<p> 3) The etc/passwd file </p>"
                    content +="<p> The quality of the report will affect the
evaluation of the practical part.La qualita' del report incidera' sulla
valutazione della parte pratica.</p>"
            else:
                code = httplib.NOT_FOUND
        except Exception, ex:
            content = ex.output if isinstance(ex, subprocess.CalledProcessError)
else traceback.format_exc()
            code = httplib.INTERNAL_SERVER_ERROR
        finally:
            self.send_response(code)
            self.send_header("Connection", "close")
            self.send_header("X-XSS-Protection", "0")
                self.send_header("Content-Type", "%s%s" % ("text/html" if
content.startswith("<!DOCTYPE html>") else "text/plain", "; charset=%s" %
params.get("charset", "utf8")))
            self.end_headers()
            self.wfile.write("%s%s" % (content, HTML_POSTFIX if HTML_PREFIX
in content and GITHUB not in content else ""))
```

```
            self.wfile.flush()
            self.wfile.close()

class ThreadingServer(SocketServer.ThreadingMixIn, BaseHTTPServer.HTTPServer):
    pass

if __name__ == "__main__":
    init()
        print "Web Application executed succesfully, connect to http://localhost:8000"
    try:
                    ThreadingServer((LISTEN_ADDRESS, LISTEN_PORT), ReqHandler).serve_forever()
    except KeyboardInterrupt:
        os._exit(1)
```

**Web_SQLi_Challenge.py**

```
#!/usr/bin/env python
import BaseHTTPServer, cgi, cStringIO, httplib, json, os, pickle, random, re, SocketServer, sqlite3, string, sys, subprocess, time, traceback, urllib, xml.etree.ElementTree
try:
    import lxml.etree
except ImportError:
    print "[!] If you want to run it locally install 'python-lxml' (e.g. '%s')\n" % ("apt install python-lxml" if not subprocess.mswindows else "https://pypi.python.org/pypi/lxml")

NAME, VERSION, GITHUB, AUTHOR, LICENSE = "Sec Lab Exam 11 June (2021) < SQLi challenge (Login Bypass)", "0.01", "https://gitlab.com/wild_boar/labsec_course", "@wild_boar", "GPLv3"
LISTEN_ADDRESS, LISTEN_PORT = "0.0.0.0", 8000
HTML_PREFIX, HTML_POSTFIX = "<!DOCTYPE html>\n<html>\n<head>\n<style>a {font-weight: bold; text-decoration: none; visited: blue; color: blue;} ul {display: inline-block;} .disabled {text-decoration: line-through; color: gray} .disabled a {visited: gray; color: gray; pointer-events: none; cursor: default} table {border-collapse: collapse; margin: 12px; border: 2px solid black} th, td {border: 1px solid black; padding: 3px} span {font-size: larger; font-weight: bold}</style>\n<title>%s</title>\n</head>\n<body style='font: 12px monospace'>\n<script>function process(data) {alert(\"Surname(s) from JSON results: \" + Object.keys(data).map(function(k) {return data[k]}));}; var index=document.location.hash.indexOf('lang='); if (index != -1) document.write('<div style=\"position: absolute; top: 5px; right: 5px;\">Chosen language: <b>' + decodeURIComponent(document.location.hash.substring(index + 5)) + '</b></div>');</script>\n" % cgi.escape(NAME), "<div style=\"position:
```

```
fixed; bottom: 5px; text-align: center; width: 100%%;\">Powered by <a
href=\"%s\" style=\"font-weight: bold; text-decoration: none; visited: blue;
color: blue\" target=\"_blank\">%s</a>
(v<b>%s</b>)</div>\n</body>\n</html>" % (GITHUB, "@wild_boar",
VERSION)
USERS_XML = """<?xml version="1.0" encoding="utf-8"?><users><user
id="0"><username>admin</username><name>admin</name><surname
>admin</surname><password>7en8aiDoh!</password></user></users>
"""

def init():
    global connection
        connection = sqlite3.connect(":memory:", isolation_level=None,
check_same_thread=False)
    cursor = connection.cursor()
        cursor.execute("CREATE TABLE users(id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT, name TEXT, surname TEXT, password
TEXT)")
    cursor.executemany("INSERT INTO users(id, username, name, surname,
password) VALUES(NULL, ?, ?, ?, ?)", ((_.findtext("username"),
_.findtext("name"), _.findtext("surname"), _.findtext("password")) for _ in
xml.etree.ElementTree.fromstring(USERS_XML).findall("user")))
     cursor.execute("CREATE TABLE comments(id INTEGER PRIMARY KEY
AUTOINCREMENT, comment TEXT, time TEXT)")

class ReqHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_GET(self):
        path, query = self.path.split('?', 1) if '?' in self.path else (self.path, "")
            code, content, params, cursor = httplib.OK, HTML_PREFIX,
dict((match.group("parameter"),
urllib.unquote(','.join(re.findall(r"(?:\A|[?&])%s=([^&]+)"
match.group("parameter"), query))))
re.finditer(r"((\A|[?&])(?P<parameter>[\w\[\]]+)=)([^&]+)",
connection.cursor()
        try:
            if path == '/':
                if "id" in params:
                    cursor.execute("SELECT id, username, name, surname FROM
users WHERE id=" + params["id"])
                content +=
"<div><span>Result(s):</span></div><table><thead><th>id</th><th>
username</th><th>name</th><th>surname</th></thead>%s</table>
%s" % ("".join("<tr>%s</tr>" % "".join("<td>%s</td>" % ("-" if _ is None
else _) for _ in row) for row in cursor.fetchall()), HTML_POSTFIX)
                elif "v" in params:
                    content += re.sub(r"(v<b>)[^<]+(</b>)", r"\g<1>%s\g<2>"
% params["v"], HTML_POSTFIX)
                elif "object" in params:
                    content = str(pickle.loads(params["object"]))
                elif "path" in params:
                    content = (open(os.path.abspath(params["path"]), "rb") if not
"://" in params["path"] else urllib.urlopen(params["path"])).read()
```

```
            elif "domain" in params:
                content = subprocess.check_output("nslookup " + params["domain"], shell=True, stderr=subprocess.STDOUT, stdin=subprocess.PIPE)
            elif "xml" in params:
                content = lxml.etree.tostring(lxml.etree.parse(cStringIO.StringIO(params["xml"]), lxml.etree.XMLParser(no_network=False)), pretty_print=True)
            elif "name" in params:
                found = lxml.etree.parse(cStringIO.StringIO(USERS_XML)).xpath(".//user[name/text()='%s']" % params["name"])
                content += "<b>Surname:</b> %s%s" % (found[-1].find("surname").text if found else "-", HTML_POSTFIX)
            elif "size" in params:
                start, _ = time.time(), "<br>".join("#" * int(params["size"]) for _ in range(int(params["size"])))
                content += "<b>Time required</b> (to 'resize image' to %dx%d): %.6f seconds%s" % (int(params["size"]), int(params["size"]), time.time() - start, HTML_POSTFIX)
            elif "comment" in params or query == "comment=":
                if "comment" in params:
                    cursor.execute("INSERT INTO comments VALUES(NULL, '%s', '%s')" % (params["comment"], time.ctime()))
                    content += "Thank you for leaving the comment. Please click here <a href=\"/?comment=\">here</a> to see all comments%s" % HTML_POSTFIX
                else:
                    cursor.execute("SELECT id, comment, time FROM comments")
                    content += "<div><span>Comment(s):</span></div><table><thead><th>id</th><th>comment</th><th>time</th></thead>%s</table>%s" % ("".join("<tr>%s</tr>" % "".join("<td>%s</td>" % ("-" if _ is None else _) for _ in row) for row in cursor.fetchall()), HTML_POSTFIX)
            elif "include" in params:
                backup, sys.stdout, program, envs = sys.stdout, cStringIO.StringIO(), (open(params["include"], "rb") if not "://" in params["include"] else urllib.urlopen(params["include"])).read(), {"DOCUMENT_ROOT": os.getcwd(), "HTTP_USER_AGENT": self.headers.get("User-Agent"), "REMOTE_ADDR": self.client_address[0], "REMOTE_PORT": self.client_address[1], "PATH": path, "QUERY_STRING": query}
                exec(program) in envs
                content += sys.stdout.getvalue()
                sys.stdout = backup
            elif "redir" in params:
                content = content.replace("<head>", "<head><meta http-equiv=\"refresh\" content=\"0; url=%s\"/>" % params["redir"])
            if HTML_PREFIX in content and HTML_POSTFIX not in content:
                content += "<div><span>Web Challenge Exam 11 June 2021:</span></div>\n<p> The challenge is to exploit a sql injection to bypass the login</p><p> The login is done via a GET call to the address
```

```
/login with parameters username and password e.g. /login?username=&password=</p><p> The purpose of the exercise is to log in as an admin user </p><p> To get the exercise correct, the student must hand in: the FLAG, the complete query that exploits the SQLi, a brief description of the steps taken, why the vulnerability exists and how it is exploited. </p><p> Delivering only the FLAG will not be considered sufficiente.</p>"
                    #\n<ul>%s\n</ul>\n" % ("".join("\n<li%s>%s - <a href=\"%s\">vulnerable</a>|<a href=\"%s\">exploit</a>|<a href=\"%s\" target=\"_blank\">info</a></li>" % (" class=\"disabled\" title=\"module 'python-lxml' not installed\"" if ("lxml.etree" not in sys.modules and any(_ in case[0].upper() for _ in ("XML", "XPATH"))) else "", case[0], case[1], case[2], case[3]) for case in CASES)).replace("<a href=\"None\">vulnerable</a>|", "<b>-</b>|")
            elif path == "/users.json":
                content = "%s%s%s" % ("" if not "callback" in params else "%s(" % params["callback"], json.dumps(dict((_.findtext("username"), _.findtext("surname")) for _ in xml.etree.ElementTree.fromstring(USERS_XML).findall("user"))), "" if not "callback" in params else ")")
            elif path == "/login":
                cursor.execute("SELECT * FROM users WHERE username='" + re.sub(r"[^\w]", "", params.get("username", "")) + "' AND password='" + params.get("password", "") + "'")
                content += "Welcome Student Here is your flag: SEC{Login_Bypass_Always_Escape_Your_Character_Come_se_fosse_antani_need_a_longer_query}<b>%s</b><meta http-equiv=\"Set-Cookie\" content=\"SESSIONID=%s; path=/\"><meta http-equiv=\"refresh\" content=\"1; url=/\"/>" % (re.sub(r"[^\w]", "", params.get("username", "")), "".join(random.sample(string.letters + string.digits, 20))) if cursor.fetchall() else "The username and/or password is incorrect<meta http-equiv=\"Set-Cookie\" content=\"SESSIONID=; path=/; expires=Thu, 01 Jan 1970 00:00:00 GMT\">"
            else:
                code = httplib.NOT_FOUND
        except Exception, ex:
            content = ex.output if isinstance(ex, subprocess.CalledProcessError) else traceback.format_exc()
            code = httplib.INTERNAL_SERVER_ERROR
        finally:
            self.send_response(code)
            self.send_header("Connection", "close")
            self.send_header("X-XSS-Protection", "0")
            self.send_header("Content-Type", "%s%s" % ("text/html" if content.startswith("<!DOCTYPE html>") else "text/plain", "; charset=%s" % params.get("charset", "utf8")))
            self.end_headers()
            self.wfile.write("%s%s" % (content, HTML_POSTFIX if HTML_PREFIX in content and GITHUB not in content else ""))
            self.wfile.flush()
            self.wfile.close()
```

```
class ThreadingServer(SocketServer.ThreadingMixIn, BaseHTTPServer.HTTPServer):
    pass

if __name__ == "__main__":
    init()
    print "%s #v%s\n by: %s\n\n[i] running HTTP server at '%s:%d'..." % (NAME, VERSION, AUTHOR, LISTEN_ADDRESS, LISTEN_PORT)
    try:
        ThreadingServer((LISTEN_ADDRESS, LISTEN_PORT), ReqHandler).serve_forever()
    except KeyboardInterrupt:
        os._exit(1)
```

**Web_Path_Traversal_Challenge.py**

```
#!/usr/bin/env python
import BaseHTTPServer, cgi, cStringIO, glob, httplib, json, os, pickle, random, re, SocketServer, sqlite3, string, sys, subprocess, time, traceback, urllib, xml.etree.ElementTree
try:
    import lxml.etree
except ImportError:
    print "[!] Successful execution connected to http://localhost:8000 \n If you want to run it install 'python-lxml' (e.g. '%s')\n " % ("apt install python-lxml" if not subprocess.mswindows else "https://pypi.python.org/pypi/lxml")

NAME, VERSION, GITHUB, AUTHOR, LICENSE = "Sec Lab Exam 25 June (2021) < Path Traversal Challenge", "0.01", "https://gitlab.com/wild_boar/labsec_course", "@wild_boar", "GPLv3"
LISTEN_ADDRESS, LISTEN_PORT = "0.0.0.0", 8000
HTML_PREFIX, HTML_POSTFIX = "<!DOCTYPE html>\n<html>\n<head>\n<style>a {font-weight: bold; text-decoration: none; visited: blue; color: blue;} ul {display: inline-block;} .disabled {text-decoration: line-through; color: gray} .disabled a {visited: gray; color: gray; pointer-events: none; cursor: default} table {border-collapse: collapse; margin: 12px; border: 2px solid black} th, td {border: 1px solid black; padding: 3px} span {font-size: larger; font-weight: bold}</style>\n<title>%s</title>\n</head>\n<body style='font: 12px monospace'>\n<script>function process(data) {alert(\"Surname(s) from JSON results: \" + Object.keys(data).map(function(k) {return data[k]}));}; var index=document.location.hash.indexOf('lang='); if (index != -1) document.write('<div style=\"position: absolute; top: 5px; right: 5px;\">Chosen language: <b>' + decodeURIComponent(document.location.hash.substring(index + 5)) + '</b></div>');</script>\n" % cgi.escape(NAME), "<div style=\"position: fixed; bottom: 5px; text-align: center; width: 100%%;\">Powered by <a href=\"%s\" style=\"font-weight: bold; text-decoration: none; visited: blue; color: blue\" target=\"_blank\">%s</a>
```

```
(v<b>%s</b>)</div>\n</body>\n</html>" % (GITHUB, "@wild_boar",
VERSION)
USERS_XML = """<?xml version="1.0" encoding="utf-8"?><users><user
id="0"><username>admin</username><name>admin</name><surname
>admin</surname><password>7en8aiDoh!</password></user></users>
"""

def init():
    global connection
        connection = sqlite3.connect(":memory:", isolation_level=None,
check_same_thread=False)
    cursor = connection.cursor()
        cursor.execute("CREATE TABLE users(id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT, name TEXT, surname TEXT, password
TEXT)")
    cursor.executemany("INSERT INTO users(id, username, name, surname,
password) VALUES(NULL, ?, ?, ?, ?)", ((_.findtext("username"),
_.findtext("name"), _.findtext("surname"), _.findtext("password")) for _ in
xml.etree.ElementTree.fromstring(USERS_XML).findall("user")))
     cursor.execute("CREATE TABLE comments(id INTEGER PRIMARY KEY
AUTOINCREMENT, comment TEXT, time TEXT)")

class ReqHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_GET(self):
        path, query = self.path.split('?', 1) if '?' in self.path else (self.path, "")
            code, content, params, cursor = httplib.OK, HTML_PREFIX,
dict((match.group("parameter"),
urllib.unquote(','.join(re.findall(r"(?:\A|[?&])%s=([^&]+)"
match.group("parameter"), query))))
re.finditer(r"((\A|[?&])(?P<parameter>[\w\[\]]+)=)([^&]+)",
connection.cursor()
        try:
            if path == '/':
                if "path" in params:
                    paths=str(params["path"])
                    if re.search("^\.\./",paths):
                        content = "Error! Can't invoke a path starting with ../, are
you sure it's a path traversal?"
                    elif re.search("^/\.\.",paths):
                        content = "Error! It is not possible to invoke a path starting
with /.., are you sure it is a traversal path?"
                    elif re.search("^//",paths):
                content = "Error! It's also not possible to invoke a path with //,
maybe you can make something newest and combination of both?"
                elif re.search("^/[a-zA-Z0-9]+",paths):
                        content = "Error! We have disabled the ability to directly
invoke the /etc/passwd file, the path parameter cannot start with / followed
by any alphanumeric character. HINT: / + alphanumeric character, what is
left out?"
                    elif re.search("wd$",paths):
                        content = "Error! The path cannot end with \"wd\", what can
you use to overide this +wild+ filter?"
```

```
                else:
                    print glob.glob(paths)
                    final = glob.glob(paths)[0]
                    print final
                    content = (open(os.path.abspath(final), "rb") if not "://" in
final else urllib.urlopen(final)).read()
            elif "redir" in params:
                content = content.replace("<head>", "<head><meta http-
equiv=\"refresh\" content=\"0; url=%s\"/>" % params["redir"])
            if HTML_PREFIX in content and HTML_POSTFIX not in content:
                content +="<div><span>Web Challenge EXam 25 June
2021:</span></div>\n"
                content +="<p> The challenge is to exploit a LFI (Local File
Inclusion) </p>"
                content +="<p> Through a GET request to the root / with the
path parameter it is possible to open a local file e.g.
/?path=file_locale.txt</p>"
                content +="<p> The purpose of this exercise is to exploit the
LFI and retrieve/read the file /etc/passwd</p>"
                content +="<p> There are some filters on the characters that
you can send as a path</p>"
                content +="<p> For each filter \"matched\" a small hint will be
proposed</p>"
                content +="<p> The student must the submit 3 files</p>"
                content +="<p> 1) The file report.txt where he/she should
explain in a understandable way the concept behind the vulnerability of this
challenge</p>"
                content +="<p> The student should also list the various steps
and attempts made to exploit the vulnerability, with the reasoning behind
bypasssing the filters, including of course the final payload</p>"
                content +="<p> 2) The screenshot where you can clearly see
the call to the application with the payload and the final result  </p>"
                content +="<p> 3) The etc/passwd file </p>"
                content +="<p> The quality of the report will affect the
evaluation of the practical part.</p>"
            else:
                code = httplib.NOT_FOUND
        except Exception, ex:
            content = ex.output if isinstance(ex, subprocess.CalledProcessError)
else traceback.format_exc()
            code = httplib.INTERNAL_SERVER_ERROR
        finally:
            self.send_response(code)
            self.send_header("Connection", "close")
            self.send_header("X-XSS-Protection", "0")
            self.send_header("Content-Type", "%s%s" % ("text/html" if
content.startswith("<!DOCTYPE html>") else "text/plain", "; charset=%s" %
params.get("charset", "utf8")))
            self.end_headers()
            self.wfile.write("%s%s" % (content, HTML_POSTFIX if HTML_PREFIX
in content and GITHUB not in content else ""))
            self.wfile.flush()
```

```
            self.wfile.close()

class                              ThreadingServer(SocketServer.ThreadingMixIn,
BaseHTTPServer.HTTPServer):
    pass

if __name__ == "__main__":
    init()
     print "%s #v%s\n by: %s\n\n[i] The execution was succesfull you can
connect  via  browser  to'%s:%d'..."  %  (NAME,  VERSION,  AUTHOR,
LISTEN_ADDRESS, LISTEN_PORT)
    try:
                          ThreadingServer((LISTEN_ADDRESS,     LISTEN_PORT),
ReqHandler).serve_forever()
    except KeyboardInterrupt:
        os._exit(1)
```