# Projeto Segurança Informática em Redes e Sistemas

THE CORK, GROUP A57

96902 | Pedro Gomes
100740 | João Salgueiro
105688 | Tiago Figueiredo

6 de janeiro de 2023

# I. Business Context

TheCork is a service that allows its users to taste the best food and wines in town by enabling easy restaurant reservations via a very user-friendly mobile application that integrates seamlessly with the restaurants' calendars.

TheCork offers a mobile app to its customers to display the list of restaurants available at the user's location. It also allows them to book a table for a specific number of people if the restaurant still has free tables. There is also a website that provides the same functionality.

TheCork also provides back-office for the restaurants which is used to manage the schedule and to approve or deny the customers reservations.

# II. Infrastructure Overview

The group decided to implement an infrastructure in OS Ubuntu v22.04. There are two virtual networks, between the CORKWEB VM and the firewall VM and the CORKDB VM and the firewall VM. The CORKWEB VM will work as a web application server being the service provider and the CORKDB VM will work as a database server. The firewall VM will serve as a gateway between CORKWEB and CORKDB and will control communications between them. It will also check and allow/disallow traffic coming to CORKWEB from the internet.

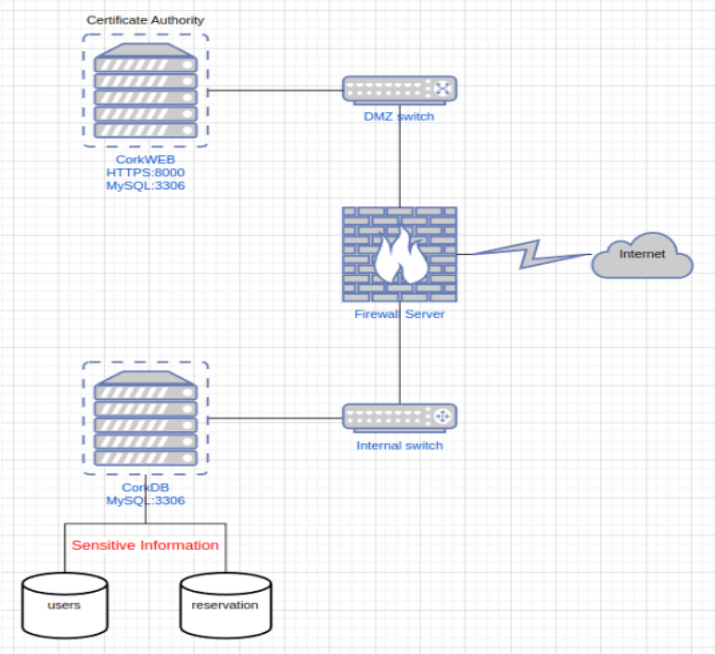


*Figure 1 - Network diagram*

| # Interface | Subnet | Adapter |
|---|---|---|
| **CORKWEB:** | | |
| 1 | 192.168.0.254 | enp0s8 |
| **FIREWALL:** | | |
| 1 | 192.168.0.5 | enp0s8 |
| 2 | 192.168.1.5 | enp0s9 |
| 3 | INTERNET | enp0s3 |
| **CORKDB:** | | |
| 1 | 192.168.2.4 | enp0s3 |

*Figure 2 - Network Configuration*

For the database server, MySQL was installed, and an instance called cork was created, with its own admin user "corkadmin". That instance will hold all the Tables for this project, as seen below.

To ease operation of the database server, the GUI MySQL Workbench was also installed.

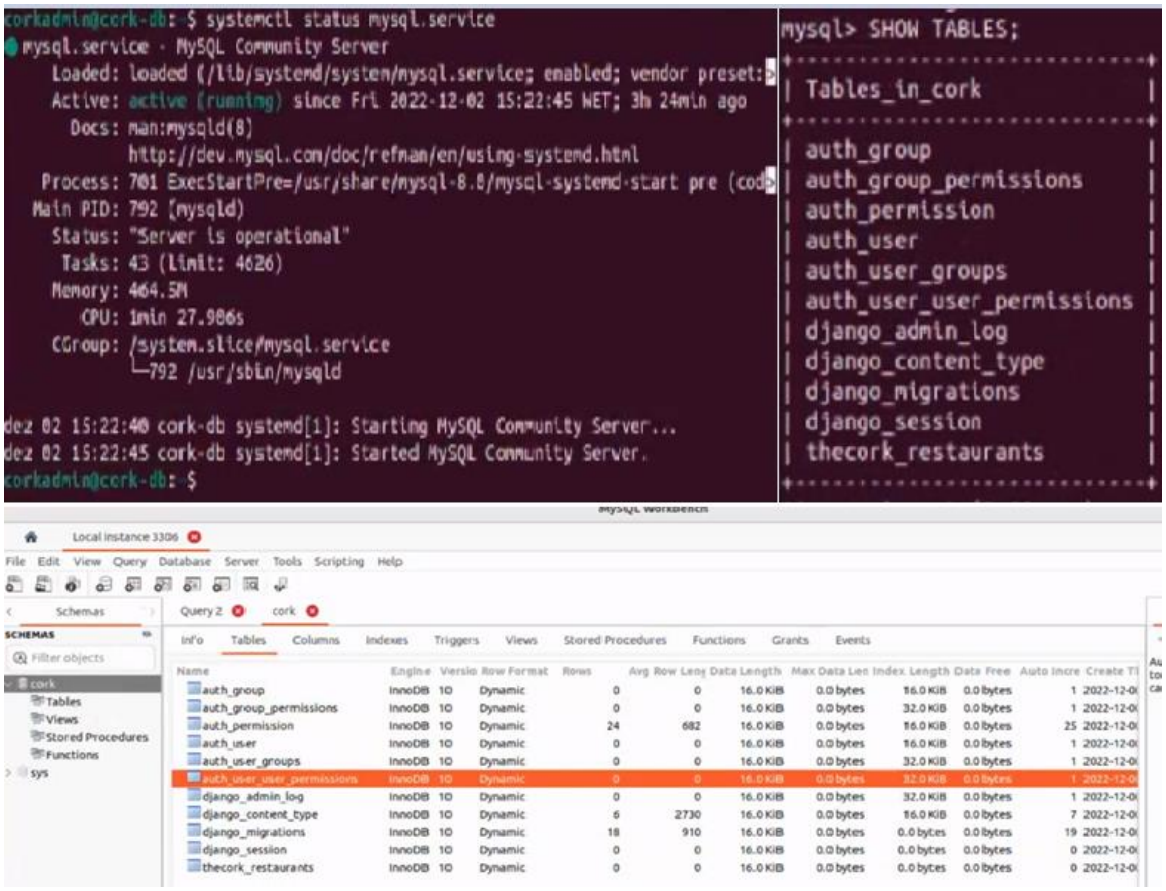


*Figure 3 – Database details*

To implement the Web App, the group decided to use a framework called Django. That framework uses Python as a programming language and is structured on a design pattern called MVT (Model-View-Template) while providing a lot of custom built in administration tools for user management along with the multitude of Python libraries which could prove necessary for this project.

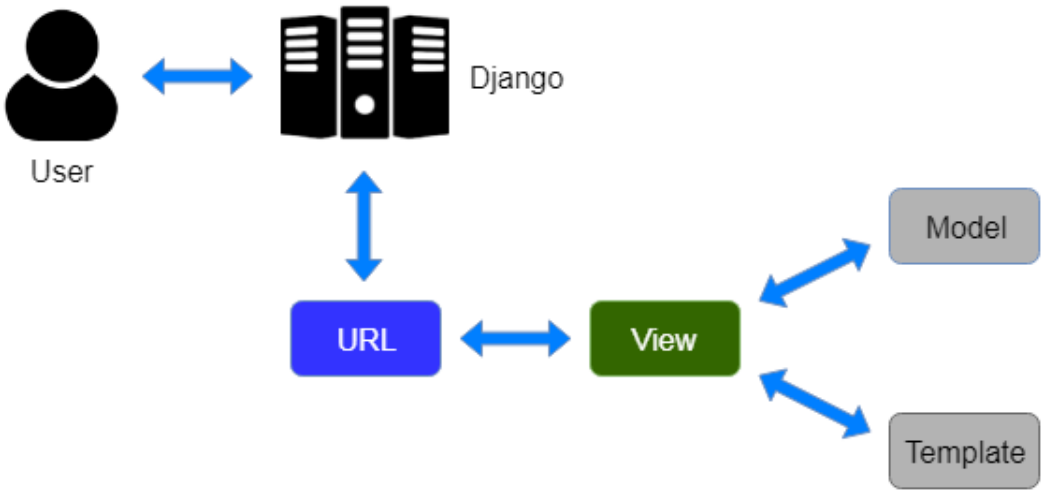


*Figure 4 - MVT Pattern*

The template part handles the user interface using html pages, the view executes all the logic and interacts with the model to carry data and the model handles the relational structure implemented in the database.

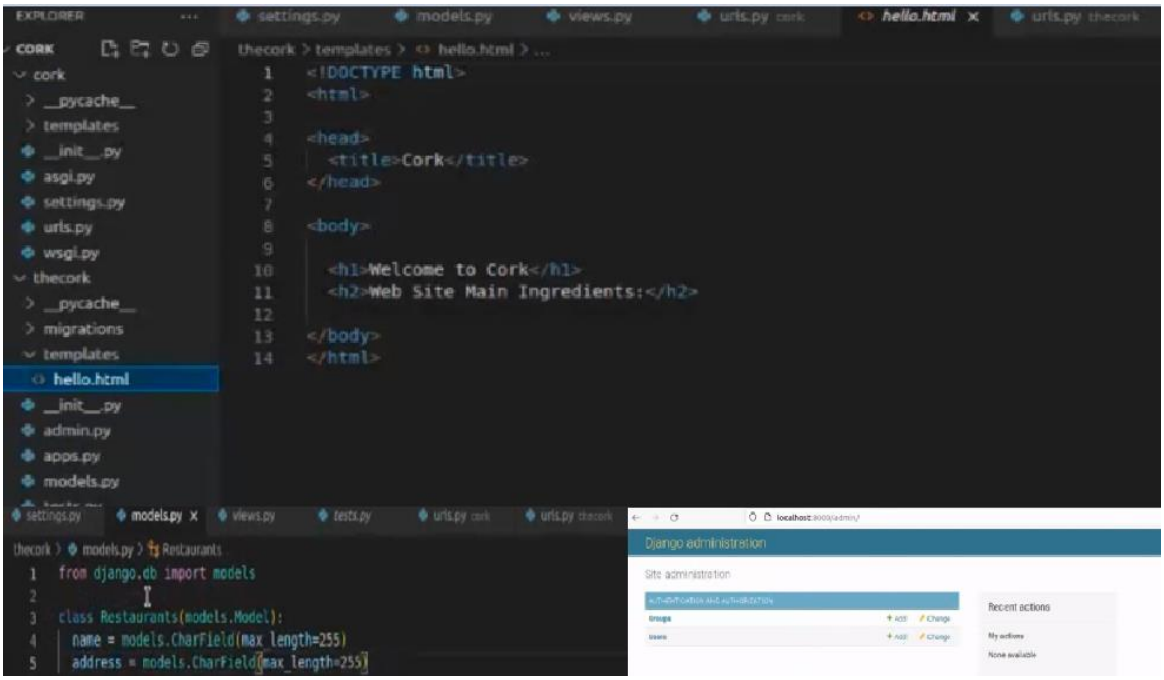


*Figure 5 - Django Overview*

# III. Secure communications

## SECURE CHANNELS

1. All communication between the users (External Networks) and the WebServer (DMZ) is forwarded through the firewall. The webserver only accepts HTTPS connections on the defined port. The firewall is set up as to only allow forwarding of external packages to the DMZ. A TCP session is established between the user and the WebServer through the firewall. After that, all data exchanges on the application layer are shown encrypted inside TLS packets. Below there is a capture, on the firewall, of a login made from a host machine simulating a client.

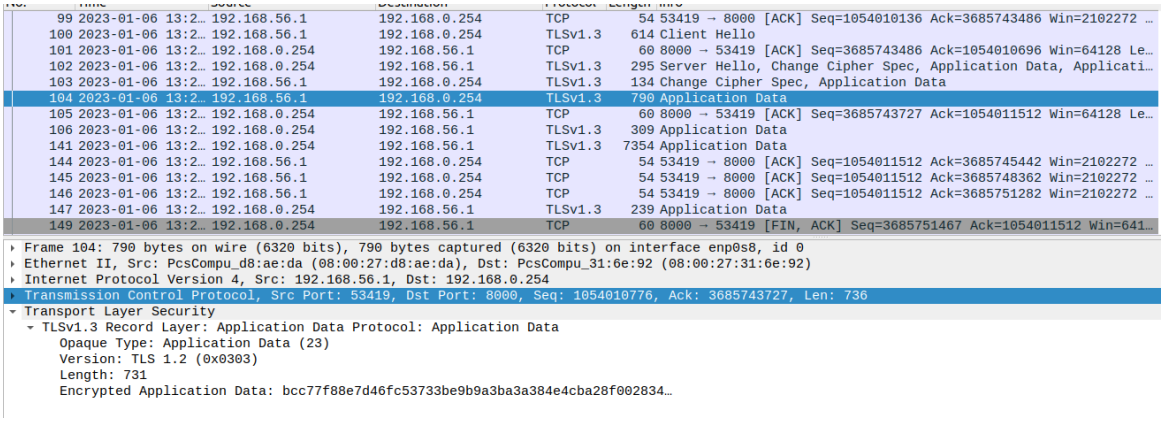


*Figure 6 – Firewall capture of Login bet. Client (192.168.56.1) and Webserver (192.168.0.254)*

2. Between the WebServer and the DatabaseServer, a Django connector manages all database operations. The forwarding of packets between the DMZ and the Internal Network is also handled by the firewall. The firewall is set up as to only allow communication between the WebServer and the DatabaseServer on designated ports. Only CORKWEB can establish a connection with the database (on port 3306) using MySQL protocol. Below there is a capture, on the firewall, of the communication between the webserver and the database while a reservation is being made.

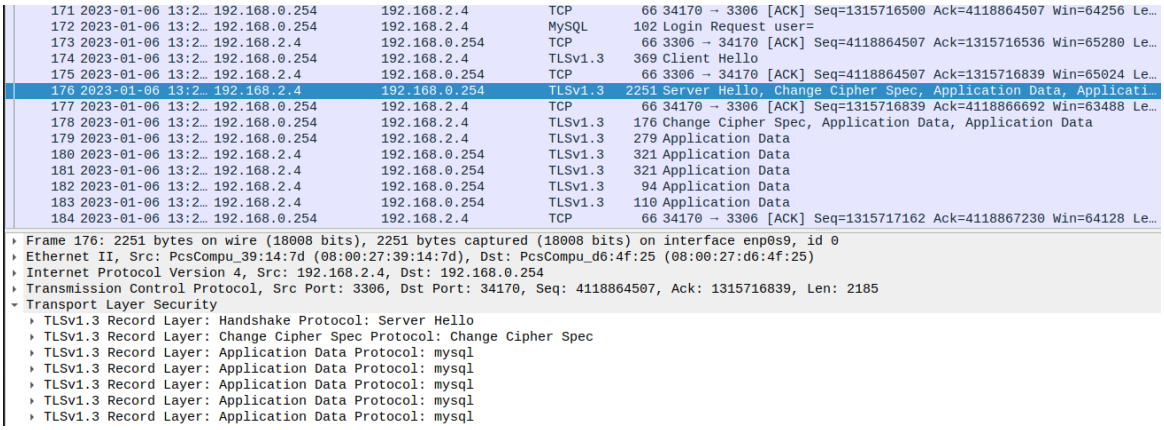


*Figure 7 -Firewall capture of DB operation bet. WebServer and DB Server (192.168.2.4)*

3. Adding to the firewall rules explained before, all other Forwards are blocked at the firewall to prevent unsecure or exploitable applications (ICMP, Netcat, etc.).

For this project, a SEED machine (similar to the ones used in the labs) was setup as a dedicated firewall. The following rules were configured on iptables:

```
$/sbin/iptables -t nat -A PREROUTING --dst 192.168.1.12 -p tcp --dport 8000 -j DNAT --to-destination 192.168.0.254:8000

$ iptables -A FORWARD -p tcp -s 192.168.2.4 --sport 3306 -d 192.168.0.254 --dport 1024:65535 -m state --state ESTABLISHED -j ACCEPT

$ iptables -A FORWARD -p tcp -s 192.168.0.254 --sport 1024:65535 -d 192.168.2.4 --dport 3306 -m state --state NEW, ESTABLISHED -j ACCEPT

$ iptables -A FORWARD -i enpos3 -o enpos8 -j ACCEPT

$ iptables -A FORWARD -o enpos3 -i enpos8 -j ACCEPT

$ iptables -A FORWARD -j DROP
```

The first rule ensures that the client browser request will be redirected to the webserver, it also masquerades the IP from the webserver. The second and third rule manage the forwards necessary for the interaction between the webserver and the database server. The fourth and fifth rule manage all external requests, by only allowing them for the DMZ. The last rule chains with the previous, to ensure no other forwards will happen.

4. To ensure HTTPS access to the Webservice, we must provide users with a certificate. The website certificate will be self-signed (we are saying that our own server is secure, which is dubious in real world situations) but serves to exemplify how a secure connection works. The CA server will provide users with certificates based on their public key and the CA private key. Our option is setting the Webserver as a certificate authority using OpenSSL.

# IV. Security Challenge

TheCork allows their users to buy Gift Cards (that may differ in cost) to offer. These Gift Cards can be then redeemed by anyone, and the respective funds will be added to the user's wallet. However, these cards may only be redeemed once. There is a need to ensure that no-one can tamper these Gift Cards to change their value, use them several times, etc.

Requirements:

1. Cards can only be used once.
2. Card value can't be changed.
3. Cards belong to a single user.
4. There can't be two equal cards.
5. Gift card codes are saved as a hash in database.

# V. Proposed solution

## APPLICATION

A login function is already implemented that ensures confidentiality and restricts the functions that each user can access. There is a Role-Based authentication system based on

two user groups: Owner (for restaurant owners, allowing CRUD operations over an Owner's Restaurants) and Client (for restaurant clients, which allow CRUD operations over a Client's Reservations). This information is stored in the database server in its own set of tables inside the application context.

One of the requisites is that no user can see other user reservations or personal information. This is guaranteed on the application side through:

-No unregistered access to the application is allowed.

-A client can only CRUD its own reservations and personal data.

-The owner of a restaurant can only CRUD his own restaurant's data and reservations made to them.

-Only the accounts marked as "staff" (currently only *corkadmin*) have access to all the information.

Following our Role-Based authentication system, there are three kinds of users: admin, Clients and Owners. The only fully trusted is *corkadmin*, which has access to Django admin console and to the database. Note that the *corkadmin* account is different from the root user of the servers. The Owner and Client groups can be considered partially trusted because they can, on different levels, interact with the system. Owner users have access to a higher number of tools, hence, can be considered on a different trust degree from the Clients. We consider not-logged users as untrusted, having no access to the site tools.

## GIFT CARD SYSTEM

Users create their own gift cards, choosing their value. When the card is created, a code will be automatically assigned to that card. The user can choose to redeem the gift card or transfer it to someone else. When doing so, the card will be deleted, and the value will be sent to the user's wallet.

Following the TLS session establishment, the user logins in his account. After logging in he will have the option to add a gift card, choosing its value. The web server will generate a code which will be delivered to the user's browser. Simultaneously, through the previously established connection with the database, the webserver will create a gift card entry to be stored. The code is only shown to the client, while being saved as a hash of that same code in the database.

When using or transferring the gift card, after login, the user will be prompted to insert the username of the card owner and the gift card code. The webserver will check the inserted code hash with the current database entries. If they match and the card was never used, you can redeem it.

ATTACKER MODEL

We will divide the attackers in two groups, Internal and External, assuming both might or not have the capacity to decrypt the network packets.

Our security measures are based on: Network Infrastructure Security with Firewall, Encryption of network packages, Role-Based Authentication System, Database Encryption (for passwords and gift cards codes).

The firewall ensures us that an external attacker can't directly access the database server/internal network. It gives few guarantees concerning the access to the DMZ, even though more specific rules could be applied (to limit access only to the WebServer to external attackers) they were not considered for this project. In the case of an internal attacker, the firewall restricts all communication between DMZ and internal network to the minimum.

The network package data encryption on TLS ensures that intercepted packets will never reveal any sensitive information to the attacker, being him internal or external, unless he is able to decrypt that data.

On the Role Based System, assuming the corkadmin credentials are never compromised, an external attacker must be registered to access the site functions. After registering, he is only able to see the information relative to him (either reservations or restaurants). He won't be able to see information regarding other users. An internal attacker, only posing as corkadmin, could be able to see all the unencrypted information present in the database.

Some sensitive information is stored encrypted on the database, guaranteeing that even as corkadmin, an internal attacker won't be able to extract another user's password or a gift card code.

# VI. Results

Requirements:

1. Cards can only be used once.

Satisfied. Whenever a gift card is redeemed by a user or transferred into another user's wallet, a flag is set, and the gift card can't be used again. This is an important aspect so a user can't redeem it more than once.

2. Card value can't be changed.

Satisfied. After generating a gift card, the value and code is set to itself, and it can't be changed. This is made to prevent a user generating a low value gift card and change it to a bigger one.

3. Cards belong to a single user.

Satisfied. The gift card is only known to the user who created it. Even if he transfers it to another user, only the gift card value will be added to his wallet, he won't know the gift card code. This makes the gift card private.

4. There can't be two equal cards.

Satisfied. Each card has a unique code to prevent a user to redeem another user's card on purpose or by accident.

5. Gift card codes are saved as a hash in database.

Satisfied. Whenever a card is generated, it's owner, value and code are stored in the database. To prevent someone with access to the database stealing the gift card, the code will be stored as a hash value. When a user redeems a card, the WebServer checks if the code matches the hash value stored and if so, adds it to the user's wallet.

# VII. Conclusion

### MAIN ACHIEVEMENTS

Through the course of this project, a lot of different technologies were applied, which gave us the opportunity to learn and or practice with a real-world case. Whether configuring a network infrastructure, configuring a firewall, developing and hosting an app and webservice, configuring a database server, configuring a certificate authority and emitting certificates a lot was learned by this group. Also, choosing between different technologies, analyzing the vulnerabilities of the designed system and the different types of possible attacks and attacker profiles gave us a better insight on information security.

### FUTURE ENHANCEMENTS

One way to improve the security of the Gift Card system would be that the one-time code be sent through another channel other than the client's Browser TCP connection. A feature could be implemented to send the code through SMS, e-mail or even physical Gift Card (generating a Manually Distrib. One-Time Password) to prevent someone from intercepting the communication and somehow deciphering the code.

Given the potential for damaging the system (creating fake restaurants, for instance), the creation of Owner users should be tied to a stronger identification system, either some sort of Third-Party OpenID or a physical documentation inspection. The same can be said for the Client users, but to a different degree. A malicious Client could try to spam fake reservations, but we considered creating fake restaurants worse for the concept of the project.

The firewall management between the external network and DMZ could be improved in order to be more restrictive. Currently it forwards all tcp requests between the outside and the DMZ.

# References

Security in Django - https://docs.djangoproject.com/en/4.1/topics/security/

Django Databases - https://docs.djangoproject.com/en/4.1/ref/databases/#mysql-notes

MySQL Protocols - https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_PROTOCOL.html

Stack Overflow - https://stackoverflow.com/