# Implementación y desarrollo de una infraestructura de Banca Online segura

David Cuadrado

**Resumen**—Hoy en día, la gente utiliza mucho Internet debido a su fàcil acceso. Es por eso, que las grandes empreses, como los bancos, quieran ofrecer sus servicios a traves de Internet, pero también se ha de tener en cuenta algunos inconvenientes. Es sabido por todos que Internet es un lugar hostil, donde nadie garantiza al 100% que la información enviada vaya al destinatario esperado, no obstante, existen herramientas que permiten que solamente el destinatario reciba la información en un elevado porcentaje. Este proyecto consiste en el desarrollo e implementación de una infraestructura de banca onlina sencilla y conocer las diferentes herramientas de código abierto que permitan garantizar la integridad, confidencialidad y accesibilidad de dicha red.

**Palabras clave**—Banca, Ciberseguridad, Internet, IDS, Java, JSON, Nmap, Pentesting, PHP, Redes, Sockets, Suricata, TCP, Web

**Abstract**—Nowadays, Internet is commonly used due to its easy accessibility. This is the reason why companies, such as Banks, are planning to offer their services online. However, several disadvantages must be taken into account.
It is already known that the Internet is a hostile place where no-one can guarantee that the information you send would be received by the expected recipient. Fortunately, there are tools and mechanisms that can ensure the reception in most of the cases. This Project consists not only on understanding a set of security open source tools to deploy i tinto a properly secured network, but also to implement and develop a simple online.

**Index Terms**—Bank, Cybersecurity, Internet, IDS, Java, JSON, Nmap, Network, Pentesting, PHP, Sockets, Suricata, TCP, Web

## 1 Introducción

La banca online fue una idea originada en 1983 con la finalidad de ofrecer una serie de ventajas a sus clientes como por ejemplo, la capacidad de poder hacer operaciones bancarias a cualquier hora sin depender de los horarios de oficina.

No obstante, esto comporta ciertos riesgos en seguridad que puede llevar a consecuencias fatales. Un ejemplo de eso es el 6 de noviembre de 2018 [1], el banco americano HSBC confirma que hubo una fuga de datos en octubre exponiendo información personal de todos los clientes de dicho banco. Otro ejemplo es que el Banco Central Europeo en 2014 [2] fue hackeado todo y que no se revelarón las cifras de los afectados.

Es por esto que a la hora de desarrollar un servicio en la nube se han de garantizar estos tres pilares fundamentales:

- Integridad
- Confiencialidad
- Disponibilidad

Un banco ha de ser capaz de garantizar que la información guardada sea visible, modificable y accesible únicamente a personas autorizadas, en este caso, un cliente ha de tener acceso únicamente a sus cuentas.

Actualmente, la mayoría de los bancos ya cuentan con una plataforma online para poder realizar transacciones, consultar el estado de las cuentas, hasta incluso, el poder contratar servicios tales como hipotecas o planes de pensión. Esto es una gran ventaja para sus clientes debido a que prácticamente pueden realizarlo todo desde su casa.

Sin ir mas lejos la aplicación de móviles de BBVA es considerada como la mejor app del año 2018 según Forrester [3] por segunda vez debido a su interfaz intuitiva y a sus servicios seguros que generan una confianza a sus clientes finales. El banco Sabadell por su parte ofrece a sus clientes un servicio de seguridad avanzado en el caso de robo de credenciales en una cuenta.

También, está saliendo a flote, un nuevo mercado de bancos online sin oficinas y multinacionales que ofrecen los mismos servicios que un banco tradicional a través de Internet. Un ejemplo de esto es la empresa financiera N26 [4], una empresa que ofrece las mismas prestaciones que un banco tradicional, además, esta empresa cuenta con una línea de cajeros automáticos propios para sacar e ingresar dinero.

No obstante, el primero en ofrecer la capacidad de hacer transferencias seguras a través de Internet es PayPal, un sistema de pago que permite hacer compras online sin necesidad de ceder los datos financieros de una persona a una tercera empresa.

El resto del artículo está organizado de la siguiente forma. La Sección 2 presenta los objetivos de este proyecto, la Sección 3 explica el estado del arte de este proyecto, indagando en las actuales soluciones de banca online reales. La Sección 4 explica la metodología utilizada en este proyecto y como se ha planteado para garantizar que la red y la aplicación sea segura. La Seccion 5 presenta el funcionamiento del proyecto y algunos resultados de las pruebas de penetración y de los tests de estrés realizados.

## 2 Objetivos

El objetivo de este proyecto es crear una infraestructura segura y confiable para los usuarios finales, desarrollando un aplicativo web de banca, y adaptar la infraestructura a dicho aplicativo. Esto no implica el poner en marcha el servicio de banca debido a que se ha de tener en cuenta un tema legislativo, si no en conocer si a dia de hoy, con herramientas de código abierto, se puede securizar una aplicación de banca online previamente desarrollada.

## 3 Estado del arte

Actualmente, las tecnologías Open Source se consideran una de las mejores soluciones de bajo coste para levantar servicios. Sin ir mas lejos *Apache httpd Server* [5] es una de las herramientas mas utilizadas para crear servidores web. También existen herramientas también Open Source orientadas a la seguridad informática tales como *OpenSSL*[6], herramienta capaz de crear certificados que permiten verificar la identidad de un servidor, *Snort*[7] que es uno de los detectores de intrusos mas utilizados actualmente o *Iptables*[8], un cortafuegos instalado en los sistemas operativos Linux que, con una sintaxis muy básica, permite gestionar el tráfico de entrada y salida del ordenador.

No obstante, el éxito de estos programas no viene por la funcionalidad en si, si no por la inmensa comunidad que hay detrás de dichas herramientas. Sin ir mas lejos la comunidad Snort se encarga, de actualizar la base de datos de ataques conocidos. La comunidad de Iptables ha desarrollado varias plantillas con reglas para configurar el cortafuegos acorde a tus necesidades. Como añadido, debido a que la instalación de algunas de estas herramientas es compleja para un usuario de nivel medio, existen distribuciones de Linux como PFSense [9] que, utilizando herramientas de código abierto, permite crear y gestionar un router y securizarlo con detectores de intrusos o con cortafuegos.

## 4 Metodología

Para conseguir los objetivos del proyecto, se ha dividido este en 3 fases y se ha utilizado una metodología en cascada tal y como se muestra en la Figura 1.

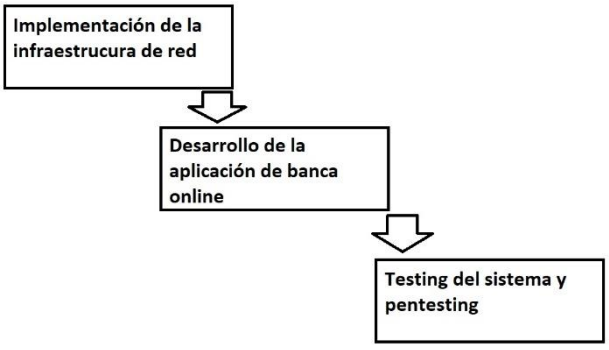


Fig. 1: Planificación de la implementación

### 4.1 Fases del proyecto

#### 4.1.1 Implementacion de la infraestructura de red

La primera fase es la implementación de la red simulando que se está desarrollando dicha red para una pequeña empresa financiera donde existen 3 subredes:

- La primera subred es llamada *DMZ* y es la única red conectada a internet.
- La segunda subred es llamada *Mainframe* y es donde está la máquina con el programa encargado de aceptar transacciones bancarias y proporcionar información de la base de datos a un *Webserver* dentro de la subred DMZ.
- La tercera subred es llamada *Maintenance* ya que es la subred donde se podrá acceder a las herramientas de monitorización y las máquinas implicadas.

Bajo esas condiciones se han configurado las subredes de la siguiente manera tal y como muestra la Figura 2.


- *E-mail de contacte: 96cuadrado@gmail.com*
- *Menció realitzada: Enginyeria de Tecnologies de la Informació.*
- *Treball tutoritzat per: Ruben (DEIC)*
- *Curs 2018/19*

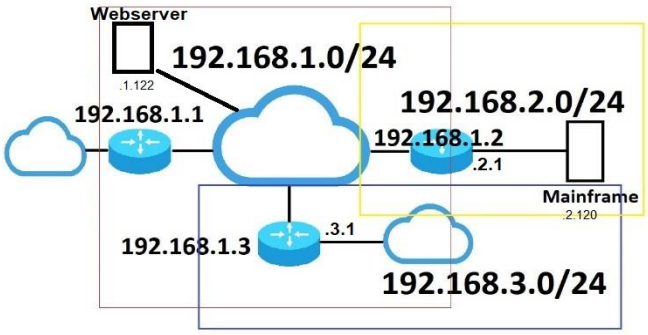


Fig. 2: Diagrama de red

El rectángulo rojo representa la subred *DMZ* con el identificador de red 192.168.1.0/24. En esa subred está situado el *WebServer* con IP estática 192.168.1.122. El gateway que conecta a INTERNET tiene como IP 192.168.1.1.

El rectángulo amarillo representa la subred *Mainframe* con la IP de red 192.168.2.0/24. El *Mainframe,* tiene como IP fija la 192.168.2.120.

El rectángulo azul representa la subred *Maintenance* con la IP de red 192.168.3.0/24. Esta red está configurada para asignar IPs mediante DHCP.

Los gateways que gestionan la red *Maintenance* utilizan el sistema operativo VyOS [10], un SO que facilita el manejo de las interfaces de redes mediante comandos simples, en cambio el router con salida a Internet utiliza PFSense, ya que, implementa un sistema detector de intrusos conocido por sus siglas IDS llamado Suricata [11]. En este proyecto no se ha tenido en cuenta cualquier otra red como la Wi-Fi o la de los empleados bancarios.

Durante esta fase también se han creado los certificados para asegurar que todas las comunicaciones dentro de la red vayan cifradas.

Los gateways están configurados de tal manera que un ordenador en la subred *DMZ*, no pueda tener acceso al resto de subredes, en cambio, un ordenador de la subred *Maintenance*, ha de tener acceso a todas las subredes. En el caso de la subred *Mainframe,* únicamente pueden acceder a dicha red el *WebServer* dentro de la red DMZ y cualquier ordenador de la subred *Maintenance*.

El cortafuegos del gateway conectado a Internet tiene las siguientes reglas mostradas en la Tabla 1 y la Tabla 2.

TABLA 1 REGLAS DE ENTRADA DEL CORTAFUEGOS

| IP | Puerto | Estado | Subred | Acción |
|---|---|---|---|---|
| ANY | ANY | NEW | INTERNET | DROP |
| ANY | 443 | NEW | INTERNET | ACCEPT |
| ANY | ANY | ESTABLISHED | INTERNET | ACCEPT |

TABLA 2 REGLAS NAT DEL CORTAFUEGOS

| srcIP | srcPort | dstIP | dstPort |
|---|---|---|---|
| ANY | ANY | INTERNET IP | ANY |
| INTERNET IP | 443 | 192.168.1.122 | 443 |

El cortafuegos del router de la subred *Mainframe* tiene las siguientes reglas mostradas en la Tabla 3 y Tabla 4.

TABLA 3 REGLAS DE ENTRADA DEL CORTAFUEGOS

| IP | Puerto | Estado | Subred | Acción |
|---|---|---|---|---|
| ANY | ANY | NEW | DMZ | DROP |
| ANY | ANY | ESTABLIS-HED | DMZ | ACCEPT |
| 192.168.1.3 | 80 | NEW | DMZ | ACCEPT |
| 192.168.1.3 | 443 | NEW | DMZ | ACCEPT |
| 192.168.1.3 | 20,21 | NEW | DMZ | ACCEPT |
| 192.168.1.3 | 22 | NEW | DMZ | ACCEPT |
| 192.168.1.122 | 5000 | NEW | DMZ | ACCEPT |

TABLA 4 REGLAS NAT DEL CORTAFUEGOS

| srcIP | srcPort | dstIP | dstPort |
|---|---|---|---|
| ANY | ANY | 192.168.1.3 | ANY |
| 192.168.1.3 | 443 | 192.168.2.120 | 443 |
| 192.168.1.3 | 20,21 | 192.168.2.120 | 20,21 |
| 192.168.1.3 | 22 | 192.168.2.120 | 22 |
| 192.168.1.122 | 5000 | 192.168.2.120 | 5000 |

El IDS se ha configurado como un sistema de prevención de intrusos. Este tipo de sistemas funcionan mediante reglas cuyo contenido son ataques previamente conocidos. Las reglas utilizadas son extraídas de un repositorio conocido como *Open Threats Exchange* [12], este repositorio es mantenido por una comunidad de usuarios. También existen reglas descargables mediante una subscripción anual, en este proyecto, este tipo de reglas no han sido contempladas.

Como añadido, se ha instalado un cortafuegos de aplicación web, también conocido como WAF en el *Web Server*, para evitar ataques conocidos como SQLi o XSS.

### 4.1.2 Desarrollo de la banca online

La segunda fase es la fase de desarrollo de una banca online simple con las siguientes operaciones:

- Login de usuario
- Registro de usuario
- Crear cuentas bancarias
- Eliminar cuentas bancarias
- Ver los movimientos de las cuentas
- Añadir intervinientes a las cuentas
- Eliminar intervinientes de las cuentas
- Hacer transferencias

La aplicación de banca online consta de 2 partes:

- El Front-End programado principalmente en PHP es una página web alojada en la máquina *WebServer* dentro de la subred *DMZ*.
- El Back-End, programado en Java, es el encargado de proporcionar información al Front-End y poder hacer operaciones tales como las transferencias.

La comunicación entre el Front-End y el Back-End se hace a través de sockets TCP y está cifrado con TLS.

Para enviar información se utiliza la siguiente estructura de datos:

`<OPCODE>|<JSON>`

El OPCODE es un número que indica de que tipo es la acción, en este caso hay 2 tipos:

- Usuario
- Transacción

El string JSON tiene diferentes propiedades en función de la operación a realizar interpretadas por ambas partes.

El número de cuenta bancaria sigue el estándar Codigo Cuenta Corriente, una numeración de 20 digitos utilizada en España para asignar los números de cuenta bamcarias, explicado en el Anexo 1.

Actualmente, la cuenta bancaria dispone de 2 saldos:
- **Saldo contable:** El el saldo que tiene una cuenta antes de ejecutar cualquier transferencia.
- **Saldo disponible:** Es el saldo que una cuenta dispone actualmente.

#### 4.1.2.1 Front-End

El Front-End está desarrollado utilizando PHP por parte del servidor, HTML5 y CSS3 para visualizar la información y AJAX como nexo entre el PHP y la interfaz web.

La página web sigue el esquema Modelo, Vista y Controlador, conocido por sus siglas MVC. Gracias a este modelo, el JavaScript irá llamando a los diferentes controladores mediante AJAX, y mostrará las diferentes vistas en función de lo que realice el cliente.

La página web se conecta al Back-End, situado en la máquina *Mainframe*, a través del modulo sockets de PHP para enviar peticiones y recibir respuesta de este. Estas comunicaciones están cifradas mediante el uso de TLS con certificados creados con *OpenSSL*.

La página web cuenta con las siguientes secciones:
- Una página de inicio de sesión para clientes registrados
- Un formulario de registro para nuevos clientes.
- Un panel de control con todas las cuentas bancarias del usuario con varias pantallas para administrarlas.

Al entrar a la página por primera vez, la primera vista es el formulario de registro para nuevos usuarios, al rellenar el formulario, se enviará un correo para verificar el correo electrónico insertado por el usuario.

Al iniciar sesión la siguiente pantalla que aparece es el panel de administración de las cuentas del usuario donde este podrá:

- Crear nuevas cuentas bancarias.
- Ver los movimientos de una cuenta específica
- Eliminar una cuenta banca
- Hacer transferencias

Para realizar las operaciones de crear cuentas y hacer transferencias es necesario una clave de autorización que se genera automáticamente cuando se inicia de sesión.

#### 4.1.2.1.1 Base de datos

La base de datos está creada para estructurar de manera ordenada la información del cliente para su fácil administración. Es una base de datos relacional utilizando el motor de bases de datos MYSQL, desarrollado por Oracle.

Las diferentes tablas que contiene la base de datos como sus relaciones vienen detalladas en la Figura 3.

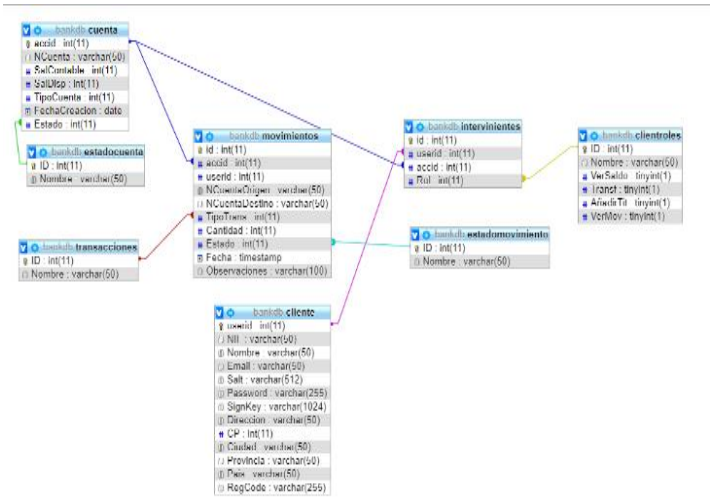


Fig. 3: Diagrama de la base de datos

La base de datos se compone de las siguientes tablas:
- **Cliente**: Tabla donde se guarda toda la información relacionada con el cliente. También guarda la contraseña de este usando el hash de BCrypt.
- **Cuenta**: Tabla donde se guarda todo lo relacionado con las cuentas bancarias.
- **Movimientos**: Tabla que relaciona el cliente y las cuentas para mostrar las operaciones que ha realizado una cuenta.
- **Intervinientes**: Tabla que relaciona el cliente y las

cuentas y muestra la gente implicada dentro de una cuenta y su rol.
- **ClientRoles:** Tabla que contiene los roles que puede tener un usuario dentro de una cuenta, en este caso existen dos: Titular y Persona Autorizada
- **EstadoMovimiento**: En esta tabla muestra los diferentes estados que puede tener una transacción, actualmente existen 3: PENDIENTE, REALIZADA y CANCELADA.
- **EstadoCuenta**: En esta tabla muestra el estado de las cuentas, en este momento solo existe el modo OPERATIVA
- **Transacciones**: En esta tabla muestra los tipos de transacciones que existen. En este caso solamente existe la transacción TRANSFERENCIA.

Para facilitar el desarrollo del Back-End en el caso de errores relacionado con el tratamiento de datos dentro de la base de datos, se ha decididio crear procedimientos almacenados, estos procedimientos almacenados son:
- **addBankAccountUser**: Este procedimiento añade un interviniente a una cuenta especificada, comprueba que la cuenta bancaria exista, que el usuario origen, es decir el usuario que agrega a otro tiene los permisos para realizar esta operación y comprueba que el usuario a insertar exista, al final de estas comprobaciones inserta el usuario en la tabla Intervinientes. El rol necesario para ejecutar este procedimiento es TITULAR.
- **changeBankAccountUserRole**: Este procedimiento cambia el rol de un usuario a otro que exista. Este procedimiento comprueba que el usuario origen tenga los permisos de TITULAR, que el usuario al cambiar el rol esté en la lista de Intervinientes de la cuenta y que el nuevo rol sea diferente al anterior que tenia.
- **createBankAccount**: Este procedimiento crea una cuenta a un usuario especifico, en este caso, se comprueba que el usuario exista y crea una nueva entrada en la tabla Cuenta y asigna al usuario que ha creado la cuenta como TITULAR de la cuenta.
- **deleteBankAccount**: Este procedimiento elimina una cuenta bancaria de un usuario. Comprueba que el usuario sea TITULAR de la cuenta, también comprueba que la cuenta no tenga ningúna transacción en estado PENDIENTE y que la cuenta no sea negativa ni tenga dinero. Al cumplirse todas las condiciones la cuenta será eliminada con todos los intervinientes de esa cuenta.
- **deleteBankAccountUser**: Procedimiento que elimina un Interviniente dentro de una cuenta. Primero comprueba que la cuenta exista, después comprueba si el que realiza la acción tiene el rol de TITULAR dentro de la cuenta y comprueba que el usuario a eliminar esté en la cuenta. Al acabar, elimina al *Interviniente*.
- **doTransaction**: Este procedimiento ejecuta una transferencia en estado PENDIENTE. Primero compueba que la transferencia exista, después comprueba si está en PENDIENTE. Una vez realizado, comprueba que haya saldo suficiente en la cuenta. Si se cumplen las condiciones realiza la transferencia descontando el saldo de una cuenta y sumándolo a la cuenta destino. Al finalizar, inserta el movimiento en las cuentas afectadas.
- **getAccountBankInfo**: Procedimiento que sirve para obtener información de una cuenta en específico. Comprueba que la cuenta exista y que el usuario tenga los permisos suficientes para poder ver la información de la cuenta. Al pasar todas las comprobaciones, envia todos los campos de la tabla Cuenta en función de la cuenta en específico.
- **getSalt**: Procedimiento almacenado que devuelve el Salt de un usuario para su posterior comprobación en el Back-End.
- **getUserBankAccounts**: Procedimiento que devuelve la lista de todas las cuentas bancarias en función del usuario.
- **insertTransactionMovement**: Procedimiento que se encarga añadir el movimiento de transacción.
- **loginUser**: Procedimiento que devuelve la información del usuario si coinciden su usuario y contraseña
- **registerUser**: Procedimiento que añade un nuevo usuario a la tabla Cliente. Comprueba si el usuario está duplicado.
- **setRegistrationCodeToUser**: Asigna un código de registro para validar el correo electrónico.
- **verifyRegistrationCodeUser**: Verifica si el código de registro dado coincide con el del usuario especifico. En caso afirmativo, valida la cuenta poniendo el campo donde se asigna el código de registro a NULL
- **setAuthCodeToUser**: Asigna un código de autorización a un usuario específico. Este procedimiento se ejecuta cuando el usuario inicia sesión.

Para la administración y gestión de la base de datos, se ha usado un cliente web llamado PHPMyAdmin [13].

#### *4.1.2.1.2 Desarrollo del Back-End*

El Back-End, está desarrollado utilizando el IDE *Eclipse Oxygen 2018-2019* [14] y consiste en un servidor encargado aceptar conexiones, procesar la información que reciba desde el Front-End y enviarle una respuesta.

##### *4.1.2.1.2.1 Estructura del proyecto*

El proyecto está estructurado utilizando la programación orientada a objetos de Java y cuenta con la siguiente estructura de clases:
- **MainframeServer**: Es la clase principal del servidor, prueba la conexión a la base de datos, crea un thread para gestionar las transacciones y escucha el puerto 5000 a la espera de conexiones.
- **UserManager**: Clase que se encarga de procesar las operaciones relacionadas con el usuario.
- **HandleConnection**: Thread que se ejecuta cuando se acepta una conexión, este thread espera la información que viene del Front-End, la formatea y devuelve una respuesta en función de la operación.
- **TransactionManager**: Clase que se encarga de gestionar todas las peticiones relacionadas con las

transacciones.

- **TransactionQueue**: Clase que se encarga de crear una cola de transacciones.
- **TransactionQueueManager**: Clase que se encarga de gestionar la cola de transacciones.
- TransactionThread: Clase que ejecuta un Thread para ejecutar las transacciones dentro de la cola
- **EmailManager**: Clase que envia correos a los clientes. En este caso solamente se envia un correo para validar el email del usuario y un correo con la clave de autorización al email del cliente. Esta clase se ejecuta en un Thread debido al elevado tiempo que tarda el correo a enviarse.
- **ServerCommands**: Clase donde está definido todos los comandos que envia de respuesta el servidor
- **UserCommands**: Clase donde están definidos todos los comandos con las operaciones que puede hacer el cliente.
- **DBModel**: Clase que se encarga de gestionar la conexión a la base de datos
- **DBController**: Clase que se encarga de realizar las diferentes consultas SQL devolviendo el resultado de estas.

Para conocer las clases implicadas al recibir un paquete del Front-End, se detalla un ejemplo del caso cuando se recibe un inicio de sesión de un cliente mostrado en la Figura 4.

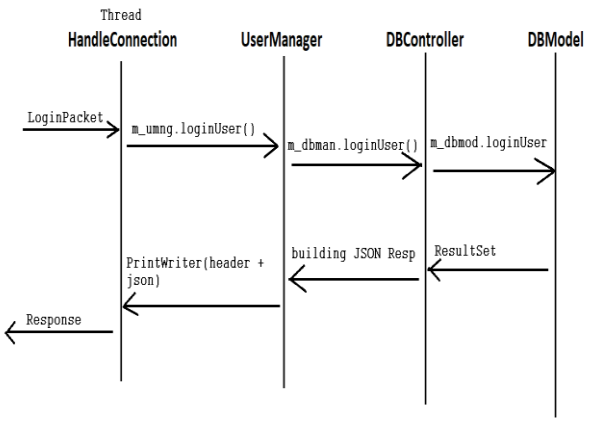


Fig 4. Diagrama de secuencia

Uno de los problemas que se presenta durante el desarrollo es a la hora de resolver conflictos entre transacciones, para esto se ha pensado en el caso de que en una cuenta A se realicen dos transacciones a la vez a diferentes cuentas de tal manera que, si se realiza una, la otra no puede realizarse. Para solucionar esto se ha procedido a crear un nuevo estado que indica que se ha recibido la transacción, pero no no se ha procesado. Una vez asignado dicho estado, se procede a añadir dicha transacción a una cola que utiliza una política FIFO tal y como se detalla en el siguiente algoritmo.

**Algorithm 1 Recibir una transacción**

```
1:  function recvTransferencia (Transacción)
2:      Transacción.Estado = PENDIENTE
3:      insertarTransacciónAlaBD (Transacción)
4:      Cola.añadir(Transacción)
5:  end function
```

Una vez insertada a la cola, existe un Thread que recoge la primera transacción de la cola y realiza, una vez realizada, se elimina de la cola y se cambia al estado REALIZADA tal y como muestra el Algoritmo 2.

**Algorithm 2 Procesar una Transacción**

```
1:  function run () //Función que ejecutara el Thread
2:    while true
3:      if Cola.hayTransacciones
4:        Transaccion:=Cola.top()
          // Realiza la transacción
          // el procedimiento almacenado
5:        MYSQL.doTransaction(Transaccion)
6:        Cola.EliminarTop()
7:      end if
8:    end while
9:  end function
```

Otro problema que se presenta a la hora de realizar esta implementación es el caso de que por cualquier causa se cerrara el programa. Esta situación hace que la cola se pierda afectando así las transferencias dentro de esta. Para solucionarlo, el programa antes de iniciar el servidor, restaurará la cola de nuevo buscando en la base de datos todas aquellas transacciones que están en estado PENDIENTE y las insertará por fecha de llegada.

Al realizar una transferencia, cuando esta tiene el estado PENDIENTE, se descuenta de la cuenta origen el saldo disponible, pero el saldo contable se deja intacto. Esto es para poder volver al saldo anterior en el caso de que dicha cuenta realice mas de una transferencia. Una vez la transferencia ha sido validada y cambiada al estado REALIZADA, se descuenta del saldo contable.

### 4.1.3. Tercera fase: Puesta en marcha y tests

Para configurar las máquinas implicadas, es necesario acceder a los ordenadores de la subred *Maintenance* y mediante SSH transferir tanto la página web como el Back-End a producción.

Una vez transferidos los ficheros necesarios, se procede a utilizar PHPMyAdmin para restaurar la base de datos al *Mainframe.*

Una vez está todo listo para ejecutarse, se ha procedido a realizar los siguientes pasos para realizar un test funcional:

- Acceder a la página web mediante la IP pública y

comprobar que funcione el HTTPS
- Registrar un nuevo usuario
- Verificar el correo electrónico de este nuevo usuario
- Crear 2 cuentas bancarias
- Hacer una transferencia de una cuenta a otra
- Eliminar una de las dos cuentas
- Cerrar sesión

Para garantizar la integridad, confidencialidad y disponibilidad del sistema, se ha realizado un test para cada uno de los tres puntos:

- Para asegurar la integridad, se enviarán peticiones mal formatadas, realizando así posibles ataques de SQL Injection utilizando la herramienta SQLMap [15] a diversos puntos de la página.
- Para asegurar la confidencialidad se esnifará el tráfico de red en diferentes puntos dentro de la red utilizando la herramienta WireShark [16] para descubrir posibles paquetes sin cifrar.
- Para asegurar la disponibilidad del sistema se ha realizado un test de estrés directamente al Back-End para descubrir la cantidad máxima de conexiones sumultaneas que este puede soportar y, realizar una gráfica para estudiar los tiempos de respuesta para un numero en concreto de conexiones.

### 4.2. Herramientas utilizadas

En este apartado, se resumen las herramientas y librerias utilizadas durante el proyecto:

- **VyOS**: Sistema operativo basado en Linux que permite una fácil gestión de red mediante comandos simples, con esta herramienta se han desarrollado los diferentes routers.
- **PFSense**: Sistema operativo basado en FreeBSD que permite configurar un router mediante una interfaz web. Con esta herramienta se ha desarrollado el router principal conectado a Internet.
- **Suricata**: Sistema de detección de intrusos y prevención que funciona mediante unas reglas. Estas reglas contienen ataques previamente conocidos y alerta si un ataque coincide con una regla. Implementado en PFSense
- **Visual Studio**: IDE de Microsoft utilizado para desarrollar la página web escrita en PHP.
- **Eclipse Java Oxygen 2018-19**: IDE utilizado para programar el Back-End encargado de realizar las transacciones y proporcionar información al *WebServer*.
- **JDBC lib**: Librería JAVA escrita por Oracle que sirve para conectarse a una base de datos MYSQL desde un programa escrito en Java.
- **BCrypt lib**: Librería Java que sirve para utilizar el algoritmo BCrypt para cifrar las contraseñas.
- **Simple JSON lib**: Librería java que sirve para crear e interpretar objetos escritos con notación JavaScript Object Notation.
- **JavaMail API**: Librería Java escrita por Oracle que sirve para enviar correos electrónicos a un servidor SMTP.
- **Bootstrap**: Librería CSS para crear diferentes estilos de manera simple.
- **MySQL**: Motor de base de datos de código abierto

## 4.2 Metodología de trabajo

La metodología utilizada durante el TFG para la realización de las diferentes fases del proyecto es una metodología iterativa tal y como se muestra en la Figura 5.

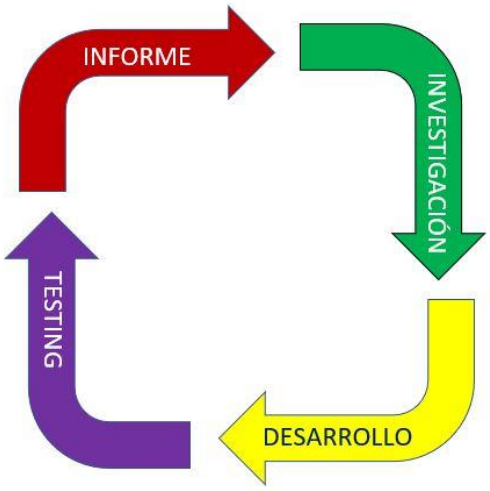


Fig. 5: Metodología de trabajo

La metodología consiste en 4 fases fundamentales:

- Investigación
- Desarrollo
- Testing
- Informe

La primera fase de la metodología es la investigación. En esta fase, se investiga como realizarse una tarea recopilando información por Internet. Esta información se clasifica para la etapada de Desarrollo

La etapa de Desarrollo consiste en ejecutar los procedimientos para acabar la tarea. Durante esta etapa se utiliza la información de la etapa Investigación para desarrollar la tarea.

La etapa de Testing se encarga de hacer pequeñas pruebas funcionales para comprobar si la tarea ha sido desarrollada correctamente. Estos pequeños tests van desde errores de sintaxi en el código fuente hasta anomalías en el flujo del programa.

La etapa de Informe se encarga de, una vez finalizado las tres etapas anteriores, se redacta un pequeño informe borrador con toda la información utilizada, los pasos ejecutados para desarrollar la tarea y los pequeños tests funcionales de la etapa Testing, para luego su posterior uso en el artículo final.

Una tarea se considera finalizada cuando esté totalmente desarrollada. En el caso de que una de las pruebas de testing funcional fallara, se volvería a empezar desde la etapa Investigación pasando primero por la etapa Informe.

## 5 RESULTADOS

Para comprobar el funcionamiento de la infraestructura de red, se ha realizado un seguimiento de flujo de paquetes desde un punto a otro mediante el comando tracepath de Linux. El resultado es el mostrado en la Figura 6


Fig. 6: Tracepath desde el Mainframe a Google

También se ha comprobado que una máquina dentro de la subred *Maintenance,* pueda conectarse mediante SSH al Back-End usando la IP del gateway que soporta la subred *Mainframe*. El resultado ha sido positivo tal y como se ilustra en la Figura 7.

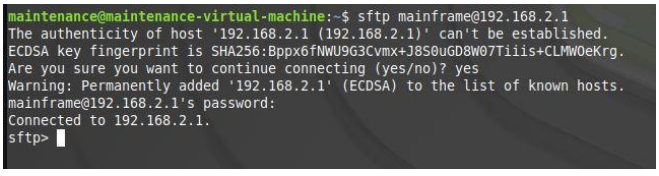


Fig 7: Conexión SSH al Mainframe desde un ordenador de la subred *Maintenance.*

Para asegurar si la red cumple con los 3 pilares en un sistema de información seguro, se han realizado los tests relacionados con la confidencialidad, integridad y disponibilidad.

Respecto a la confidencialidad, se ha decidido ejecutar el programa Wireshark para comprobar si los paquetes van cifrados. También se han instalado los certificados en el servidor Apache2 para que este funcione mediante HTTPS tal y como muestra la Figura 8.

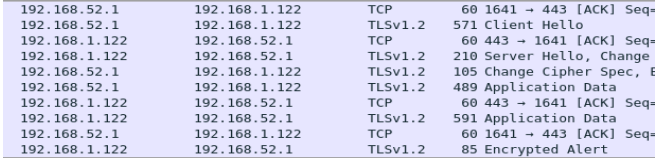


Fig. 8: Wireshark: comunicación Cliente – Front-End

La Figura 9 refleja la comunicación entre el *Front-End* y el *Back-End*. De nuevo se puede apreciar que la información que viaja va cifrada.

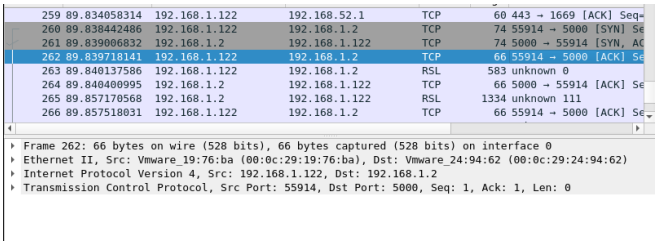


Fig. 9: Wireshark con la comunicación entre el Front-End y el Back-End

Para poner en riesgo la integridad del sistema, el atacante ha de enfocarse directamente en la aplicación web. Es por eso, que se han pensado los focos mas susceptibles a ser atacados. En este caso, se ha detectado un foco principal, la página de Inicio de sesión.

La página de Inicio seria susceptible a ataques SQLi es por eso que se ha procedido a utilizar la herramienta SQLmap para poder presenciar como reacciona el *Web Server* y el *Back-End*. El resultado se muestra en la Figura 10.

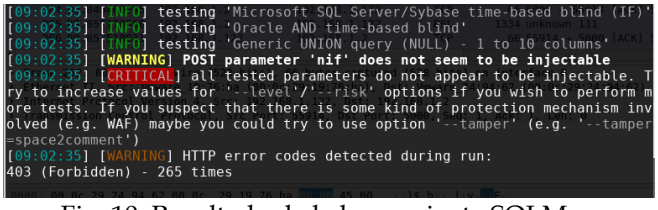


Fig. 10: Resultado de la herramienta SQLMap

El resultado es el esperado y no hay inidicios de que se haya producido ningún daño a la integridad de los datos, ya que, según la herramienta, los campos no parecen inyectables. Esto es debido a que el WAF, deniega la conexión mostrando un error 403.

Para demostrar el funcionamiento del IDS, se ha realizado un escaneo mediante el uso de la herramienta Nmap. El resultado del IDS es el que muestra a la Figura 11.

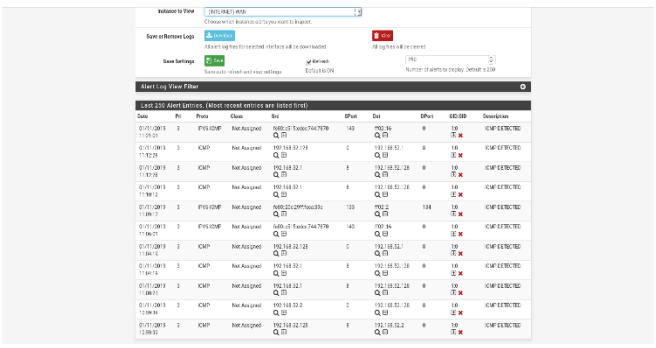

Fig. 11: Log del IDS detectando un posible ataque Nmap

Por último, para realizar el test de disponibilidad, se

ha procedido ha realizar un test de estrés tanto al Back-End como al Front-End.

Los tests de estrés se han efectuado en un ordenador con las siguientes características:

- CPU: Intel Core i7 8750H a 2.4 GHz
- RAM: 16 GB DDR4
- Disco duro: Samsung SSD M2 512 GB
- GPU: Nvidia GTX 1050ti

Para realizar el test de estrés al Back-End, se ha desarrollado un pequeño programa en JAVA que ejecuta de manera simultanea un numero de Threads que se conectán de manera concurrente al servidor y envían un paquete de LOGIN erroneo. El tiempo de conexión se calcula con la diferencia entre el tiempo de envio menos el tiempo de respuesta. Una vez los Threads han finalizado, se ha procedido a hacer la media total de todos los tiempos de conexión de cada Thread.

Los resultados del test están descritos en la Tabla 5:

TABLA 5 RESULTADOS DEL TEST DE ESTRÉS CONTRA EL BACK-END

| Cantidad de conexiones simultaneas | Tiempo en segundos |
|---|---|
| 10 | 0.223 |
| 100 | 1.43 |
| 1000 | 6.24 |
| 10000 | 64.81 |
| 100000 | 122.14 |

Durante la ejecución del test de estrés, el Back-End, ha tenido problemas al conectarse a la base de datos, rechazando conexiones debido a la inmensidad de peticiones que había por segundo. En la Figura 12 se refleja a nivel visual el resultado de la prueba.

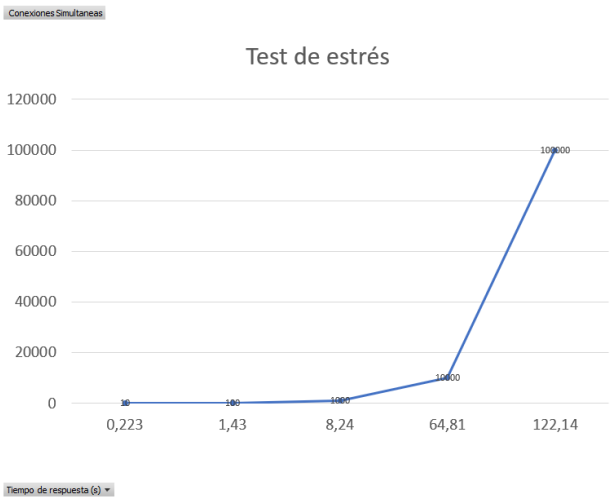


Fig. 12: Gráfica del resultado del test de estrés.

Como se puede apreciar, el tiempo de conexión crece de manera exponencial a medida que crecen el número de conexiones simultaneas.

A partir de los 10000 Threads, el sistema operativo ha empezado a rechazar conexiones, aumenta el consumo de CPU al 100% y se produce una Denegración de Servicio que deja al servidor sin poder hacer nada hasta que se procesen todas las conexiones, no obstante, se aceptan algunas conexiones.

Para realizar un test de estrés al servidor apache, que soporta al Front-End, se ha utilizado la herramienta *Apache Benchmark tool* [17] desarrollada por Apache. Esta herramienta ejecuta varios Request al servidor apache y comprueba cual de estas Request se han realizado. Se han ejecutado dos tests:

- El primer test se ejecutan 100 conexiones en 20 conexiones concurrentes.
- El segundo test, se ejecutan 5000 conexiones en 1000 conexiones concurrentes.

El resultado del primer test es mostrado en la Figura 13.

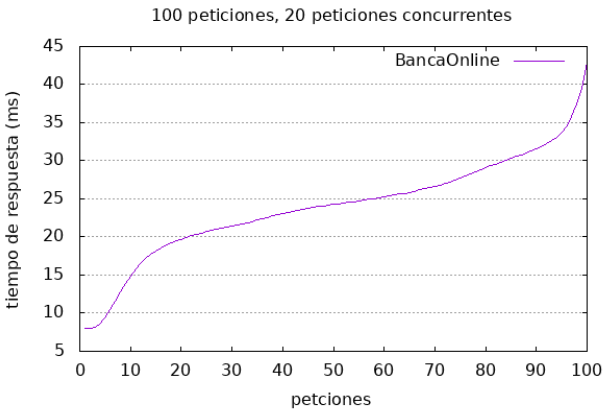


Fig. 13: Gráfica del test de estrés Apache

Esta gráfica tiene en cuenta el número de peticiones enviadas y el tiempo de respuesta. En este caso, los tiempos de respuesta alcanzan hasta aproximadamente 45 ms.
Esta situación podría considerarse normal dentro de un entorno real ya que, es posible tener 20 conexiones simultaneas. El resultado del segundo test está en la Figura 14.

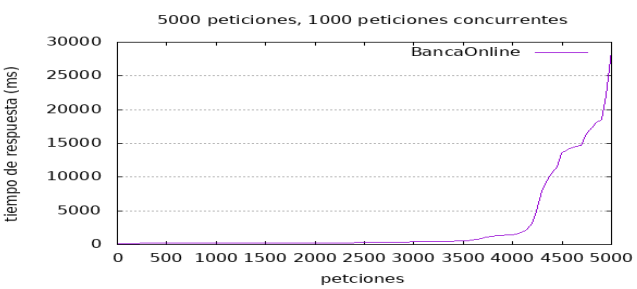


Fig. 14: Gráfica del test de estrés Apache

Siguiendo el mismo esquema que la gráfica anterior, se puede observar como los tiempos de respuesta son mayores debido a que se ejecutan de manera paralela más conexiones al servidor. Los tiempos crecen de manera exponencial hasta incluso llegar a un pico máximo de aproximadamente 27 segundos de tiempo de respuesta.

Con los tests ejecutados se ha conseguido realizar los objetivos comentados en la sección anterior pudiendo configurar correctamente la infraestructura de red y securizandola con las herramientas utilizadas.

## 6 CONCLUSION

Como conlusión al proyecto, se ha conseguido crear una infraestructura de red minimamente segura con una pequeña aplicación de banca online. En este caso, se han utilizado herramientas de código abierto y se han instalado de tal manera que cumplan una función especifica dentro de la red.

No obstante, el proyecto se ha ido desarrollando, utilizando maquinas virtuales en una máquina no especializada teniendo problemas de rendimiento. Los resultados de los tests se podrían mejorar si esta infraestructura se ejecutará en un entorno real.

Como bien se ha comentado, esta infraestructura ha sido creada utilizando herramientas de código abierto y dependiendo de una comunidad gratuita. Una posible ampliación de este proyecto es, comprarar las soluciones profesionales de empresas como FORTINET [18] o CISCO [19] para saber que ofrecen este tipo de soluciones frente a las ya utilizadas.

Por parte del desarrollo de la banca online, se ha conseguido realizar aquellas funciones mínimas que cualquier banco debería tener. Este sistema no puede ser lanzado en un entorno real debido a que hay varios puntos fuera del ámbito informático que se han de tener en cuenta, como, por ejemplo, la legislación que conlleva el crear una entidad financiera. Como ampliación del desarrollo, se podrían añadir nuevas funcionalidades tales como, productos bancarios o poder contratar varios servicios.

Para finalizar, a pesar de que se ha conseguido una mínima seguridad, esto es variable, porque no se sabe si dentro de unos meses, se detecte algún fallo de seguridad en las herramientas utilizadas que permita acceder a toda la red. Es por ello, que el administrador de redes ha de estar actualizando constantemente las máquinas implicadas.

## AGRADECIMIENTOS

Como agradecimientos, quiero agradecer a mis excompañeros del banco BBVA, especialmente a Josep Mª por explicarme en funcionamiento interno de una entidad financiera y los elementos básicos que ha de tener tales como la legislación detrás de esto.

Tambien agradecer a las diferentes personas dentro del ámbito de Ciberseguridad que me han recomendado varias herramientas de código abierto a tener en cuenta.

Por último, agradecer al tutor por ayudarme a aclarar varios puntos referentes al proyecto.

## BIBLIOGRAFIA

[1] BBC, *HBC bank confirms US data breach* https://bbc.com/news/technology-46117963 [En línea], Fecha consulta: Enero 2019
[2] CNCB. *ECB hacked: Data stolen from central bank* https://cnbc.com/2014/07/24/ecb-announces-data-theft.html [En línea], Fecha consulta: Enero 2019
[3] Forrester Research Inc. *Forrester,* https://go.forrester.com [En línea], Fecha consulta: Enero 2019
[4] N26 Inc., *N26,* https://n26.com [En línea], Fecha consulta: Enero 2019
[5] Apache Software Foundation, *Apache HTTP Server Project*. https://httpd.apache.org [En línea], Fecha consulta: Enero 2019
[6] Proyecto OpenSSL, *OpenSSL,* https://openssl.org [En Línea], Fecha consulta: Enero 2019
[7] Snort, *Snort – Network Intrusion Detection & Prevention System* https://snort.org [En línea], Fecha consulta: Enero 2019
[8] Netfilter, *Iptables* https://git.netfilter.org/iptables [En línea], Fecha consulta: Enero 2019
[9] Netgate, *PFSense,* https://pfsense.org [En línea], Fecha consulta: Enero 2019
[10] Vyos, *VyOS,* https://vyos.io [En línea], Fecha consulta: Diciembre 2018
[11] Suricata, *Suricata – Open Source IDS/ IPS/NSM engine* https://suricata-ids.org [En línea], Fecha consulta: Enero 2019
[12] AlienVault, *Open Threat Exchange,* https://alienvault.com/open-threat-exchange [En línea], Fecha consulta: Enero 2019.
[13] PHPMyAdmin Project, *PHPMyadmin,* https://phpmyadmin.net [En línea] Fecha consulta: Enero 2019
[14] The Eclipse Foundation, *Eclipse JAVA IDE,* https://eclipse.org [En Línea] Fecha consulta: Enero 2019
[15] SQLMap, *SQLMap,* https://sqlmap.org [En línea], Fecha consulta: Enero 2019
[16] Wireshark Foundation, *Wireshark go Deep*, https://wireshark.org [En línea], Fecha consulta: Enero 2019.
[17] Apache Software Foundation, *Ab,* https://httpd.apache.org [En Línea], Fecha consulta: Enero 2019
[18] Fortinet Inc, *Fortinet Security,* https://fortiner.com [En línea], Fecha consulta: Enero 2019
[19] CISCO Inc, *CISCO España,* https://cisco.com/c/es-es [En línea], Fecha de consulta: Enero 2019

# ANEXOS

## A1. CODIGO CCC

El código CCC o Codigo de Cuenta Cliente, es un código de 20 cifras impuesto por el Consejo Superior Bancario que permitia definir un estándar para identificar las cuentas corrientes.

La estructura del CCC tiene el siguiente formato:

- Los 4 primeros dígitos se corresponden al código de la entidad.
- Los 4 siguientes dígitos se corresponden al código de la oficina de origen de la cuenta bancaria
- Los 2 siguientes son los dígitos de control.
- Los 10 siguientes dígitos corresponden a la cuenta bancaria.

### A.1.1. CÁLCULO DE LOS DIGITOS DE CONTROL

El primer digito de control se calcula con el primer bloque de 8 digitos, para ello:

- Para cada digito del bloque de la entidad, se multiplica por 4, 8, 5, 10 respectivamente, al finalizar se hace el sumatorio
- Para cada digito del bloque de oficina, se multiplica por 9, 7, 3, 6 respectivamente y se suma
- Con los totales de la suma, se hace el sumario y se realiza el modulo de 11
- El digito de control es el resultado del modulo de 11 menos 11.

El segundo digito de control se calcula con los 10 digitos referente a la cuenta bancaria.

- Para cada digito del bloque, se multiplica por 1, 2, 4, 8, 10, 9, 7, 3 y 6 respectivamente y se suma
- El resultado de la suma se hace el módulo de 11.
- El digito de control es el resultado del modulo de 11 menos 11.