CS263 Coursework: Wondough Bank System

Hyun Seok Cho

March 19, 2018

1 Design

1.1 System Design

The system has been separated into 3 main components: the logic layer for the banking logic, the web servers and the front ends.

1.1.1 Logic Layer

The database stores data regarding all the bank's customers, staff members, third party merchants and logs. The data fields stored for each customer are:

- Account's username
- Account's hashed password
- Customer's email
- Customer's personal details (eg: DBO, Address, contact number, First name, Last Name)

The data fields stored for each staff member are:

- Staff member's ID
- Staff member's account hashed password
- Staff member's personal details (eg: DBO, Address, contact number, First Name, Last Name)

The data fields stored for each third party merchant are:

- Third party's name
- Third party's public key

There will be two types of logs constantly being recorded throughout the system's life-time. The first one will be the access logs: these keep a record of when a customer has logged into an account and when he/she logs out. The second type of logs will be the transaction logs: these are generated every time a customer makes a transaction.

The database admins have the highest privilege out of all three types of staff. The reason for this is because they have direct access to the database and they are in charge of making sure that the database is online and fully functional at all times. They also have access to the encryption keys used to encrypt and decrypt the files. Due to the importance of the information admins can access, they are required a separate authentication process before being able to read any data.

1.1.2 Web Servers

There are three main web servers available in the system. These are the customer web server, the staff web server and the merchant web server. These servers stand to retrieve and archive data from and into the database. For instance, when a user registrates an account, their account data needs to be stored so the front end of the customer will send this data over to the customer web server and the server will store them in the database. These servers are accessible by the developers as they must ensure that the servers store/retrieve correctly and the front ends work as expected. There

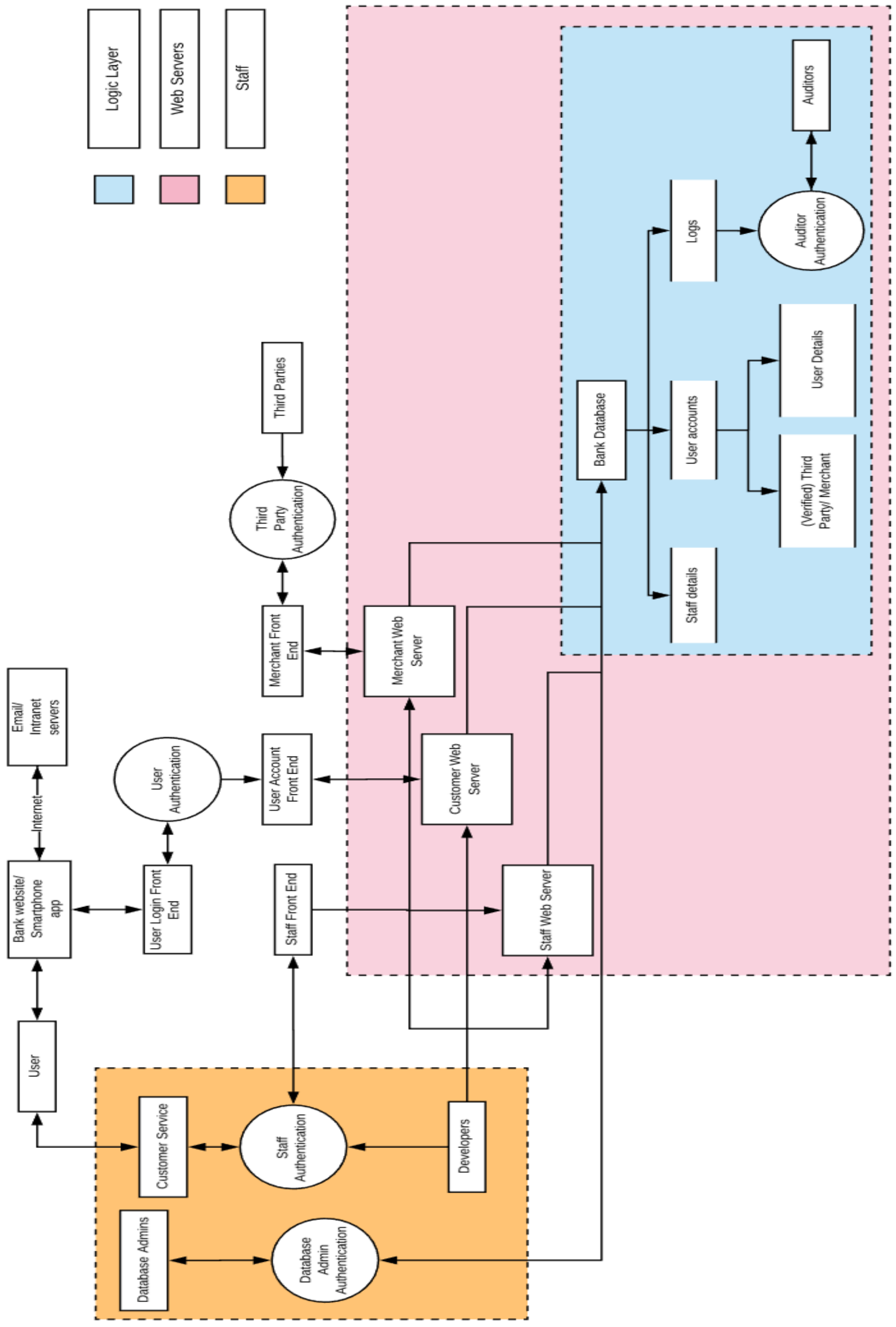


Figure 1: This is a diagram of the system design

are three occupations within the staff members: customer support service providers, developers and database-administrators. Each occupation provides different and specific privileges within the system. The customer support service providers can only gain access to a customers' data inside the database if and only if the customers themselves give permission to do so. By allowing this, the support system will be able to edit their accounts in their stead as well as make transactions for them. Note that the service providers do not have access to any of the code files.

The developers however have access to all the code files regarding the web servers and the front ends. They prioritize the functionality of the servers to maintain data consistency and integrity.

1.1.3 Front Ends

There are also three front ends: the customer front end, the staff front end and the merchant front end. These front ends are there to display the data of each individual's account and to interact with the web servers to send/retrieve data to/from the database. The front ends contain most of the authorisations required by the system by that particular entity, so for example the staff would need to authenticate themselves and after that they would gain access to the front end.

There are technically two user front ends, the user login front end and the customer front end. The user login front end is simply an initial page for customers to either login or register. In the case, they already have an account they can just login with their username and password, otherwise they are required to fill out a form to create an account. Once the customer has successfully logged in, their respective information is shown in the customer front end such as their balance, their past transactions. The customers can also contact customer service for help.

The customer support service providers can only gain access to a customers' data inside the database if and only if the customers themselves give permission to do so. By allowing this, the support system will be able to edit their accounts in their stead as well as make transactions for them. Note that the service providers do not have access to any of the code files in the web servers. These staff members have the information displayed in the staff front end.

1.2 Threat Modeling

In this subsection, possible threats to the Wondough Bank system will be identified and the counter measures applied to such situations will be examined. The threat modeling approach used for this project is "S.T.R.I.D.E".

1.2.1 Spoofing

Spoofing could occur at any point an authentication process needs to be done within our system.

- Threat: Attackers can perform brute force attacks to obtain the username and passwords of the customers.
  - Response: It is hard to prevent attackers from brute forcing the combinations but by salting all passwords and applying modern strong cryptographic hashing on them [9], the time consumption on this method can increase considerably thus attackers might not take this route to infiltrate into the system.
  - Response: Another safety measure could be to disconnect or delay their ability to attempt login in the case they fail a certain number of times consecutively. An example would be to disconnect the attacker's login front end or redirect them elsewhere, and send an alert to the corresponding email of the account asking for further verification.
- Threat: Third parties are also possible entry points for an attacker. An attacker could pose as the third party
  - Response: In order to ensure that the identity of these third parties are indeed who they say they are, digital certificates can be issued by a trusted certificate authority.
- Threat: There is also a possibility that an attacker could use a random port or socket that any of the web servers normally use and thus gaining access to the database before the client.
  - Response: To prevent such occurrences, the system can change the port numbers randomly so that the attacker can't guess which port will be used next.

- Threat: an attacker could aim to access the system through the account recovery pages as they sometimes do not require you to provide previous passwords so they might be able to simply change the account's password at will.
  - Response: To prevent potential spoofing through the account recovery pages, the old password will be required and in the case, the user is not able to provide such information then an email will be sent to the address with which the account was initially created.

1.2.2 Tampering

- Threat: The system allows the customers to change certain details about their account such as their username, password, email address, etc so someone who gets access to a machine whose account is logged in might be able to change personal data without the owner's permission.
  - Response: The current password will be required in order to edit any data or to make any transactions obviously.
- Threat: Replay attacks are a common way for attackers to fool systems into believing they are the actual customers during data exchange.
  - Response: To prevent these, tagging each encrypted component with a new session ID and component number whenever the program is ran again. These values would act in the same way as timestamps in that they stop a "man-in-the-middle" attacker from replicating the previous data to access the server.
- Threat: Since the system communicates through a website and requires users to enter data upon registration, it is likely that both HTML and JavaScript are used to implement these. The opening left when coding with these is that code injections can be introduced in the form fields.
  - Response: There are various precautions that can improve the server; firstly, all form fields can be validated to prevent invalid forms of data to be entered; secondly, the outputs can be sanitized thus filtering the submitted data before re-displaying in the browser. As a side note, the reason why the output is sanitized rather than the input is because of the third parties. By sanitizing inputs, the program might delete characters that might have significance for third parties in their own code.
- Threat: Lower privilege staff members might be able to access files they are not supposed to be able to and change their contents and possibly inject malicious codes.
  - Response: Strong file system ACLs should be set according to the different authority levels of individuals within the system. For example, developers might have access to code files but other staff members like the customer service should not.

1.2.3 Repudiation

As mentioned in the design section, logs of account accesses and transactions are kept in the database to keep track of any such activities. These by themselves provide repudiation as no one would be able to deny their actions.

- Threat: On the other hand, attacks could be launched to such logs to delete or edit such records.
  - Response: Proper ACLs should be assign to these files to prevent lower privilege attackers from tampering them.

  Unfortunately in the case a log is introduced from a legitimate account but the action was not performed by the owner of the account, then there would not be any way to tell whether the owner indeed did such actions or not.

1.2.4 Information Disclosure

- Threat: Although the files have strong ACLs as mentioned earlier, in the case they somehow ended in the attacker's hands, he/she would be able to interfere with the entire system's functionality and content.
  - Response: In order to prevent the attacker from reading such files, a safety measure would be to encrypt all data stored in the files with a standard algorithm.
- Threat: An attacker could break into the facility and steal the encryption keys or the data in the database by transferring such information to an external device.
  - Response: The encryption keys should be saved in separate machines to those where the data is stored. This is to ensure attackers from obtaining both components at the same time.
  - Response: Another important aspect of encryption keys are their life-cycles. Usually, the most secure encryption keys are set to expire after a predetermined time period so it is important to update the keys periodically and frequently to mitigate risk from a potentially stolen encryption key since they can't decrypt the data anymore without the new key.

In the earlier sections, authentication of users and third parties were briefly discussed. Another area that requires authentication are the network connections. Since the system is connected to the Internet, and in various cases a given device will require sending data over the Internet to another device eg: when a customer makes a transaction, the details of such transaction must be sent to the bank web servers.

- Threat: Network connections leave a big vulnerability to "man-in-the-middle" attacks which allow them to sniff packets and read the information being exchanged. The attacker could also impersonate either side (staff or customer) and obtain even more data by deceiving them.
  - Response: In order to know that the data from the sender has not been altered or intercepted at any point, each endpoint of the connections must have some form of authentication so that the recipient, in this case the bank web server (be it customer, staff or merchant web servers), can know that the data is valid and the sender is who he says he is and vice-versa when web servers send data back to the customer.

1.2.5 Denial of Service

- Threat: A form of denial of service would be an attacker that intentionally inserts wrong passwords and consumes the allowed number of attempts of login from an account thus triggering the disconnection of said account.
  - Response: The server can be programmed to send an alert to the email of the account so that the owner of the account can provide credentials and free his/her account.
- Threat: Another type of denial of service could be the distributed denial of service where the attacker floods the resources with excessive requests to produce massive network traffic thus overloading the system and causing it to shut-down to other users. DDOS originates from many different sources which means that prevention by blocking 1 specific source is very difficult.
  - Response: There are several approaches to ensure prevention and mitigation are covered as best as possible. Firstly, the system's resources should not be located in a single data center and should be spread out in different networks; this would prevent large network traffics from forcing a complete system failure as it is unlikely that all networks will go down at the same time.
  - Response: Secondly, the network's bandwidth could be increased to be able to withstand large volumes of traffic. The disadvantage of such method would be that it can be costly.

  - Response: Thirdly, a transfer type of solution can be performed where the system relies on an open source software to help reduce the traffic. Amongst the most-well known ones are DDoS Deflate and Fail2Ban. Some of these would require an extra fee on the system and it is up to the bank to decide whether they would like to implement these considering the likelihood of the risk and its danger.

1.2.6 Elevation of Privilege

- Threat: Someone with less privilege than an administrator could obtain the administrator details and log in to their accounts. This could result in the attacker giving privileges to other lower level accounts and thus spreading the threat across multiple users thus making the attack harder to eliminate.
  - Response: Firstly, a thorough background screening on all the administrators to check if they are truly trustworthy.
  - Response: Secondly, SID filtering could be implemented in the system. SID filtering essentially prevents domains from automatically trusting the same domains as the questionable domain. In simple terms, if domain A trusts domain B and domain B trust domain C, domain A does not necessarily have to trust domain C by transitivity if SID filtering is implemented as explained in the tech republic link.

1.3 Physical Security

In any system, there is always a threat of a physical breach occurring.

- Threat: Someone could steal a client's or staff member's account details. The risk of this event would be very dangerous because if the attacker is able to obtain the staff member's account and pretend to be that worker, then he/she might be able to fool customers into giving permission for changes in their account details thus gaining access to the staff web server and ultimately the information stored in the database could be manipulated.
  - Response: In order to prevent such events, both staff members and customers should be reminded of the risks and to be as private as possible when dealing with account details.
- Threat: An attacker could also simply break into the offices and destroy the machines containing the database, or even worse inject malicious code files into the database and original code files thus interrupting the system's functionality.
  - This kind of physical attack can be prevented by having these machines locked up in separate rooms with possibly several layers of security before reaching it such as security guards, fingerprint recognition, and mostly likely access card readers. Only the database administrators should have access to this facility.

2 Banking Legislations

Under the Data Protection Act 1998, personal data stored on computers or in an organised paper filing system must comply with the law's eight protection principles [10]. According to the law, personal information must be kept up to date which means that our system will be required to ask the users if there are any changes regularly.

All banking businesses are regulated by the Financial Services Authority and the Financial Services and Market Act (FSMA)[11] states under section 59 that a person cannot carry out certain controlling functions without the FSA's approval. Upon research, one of these functions include customer functions which involve editing or accessing any personal customer data stored on devices or organised paper without their permission. Yet another reason as to why the edition of the accounts' details is prohibited to customer service providers unless allowed to do so by the owner himself.

h!

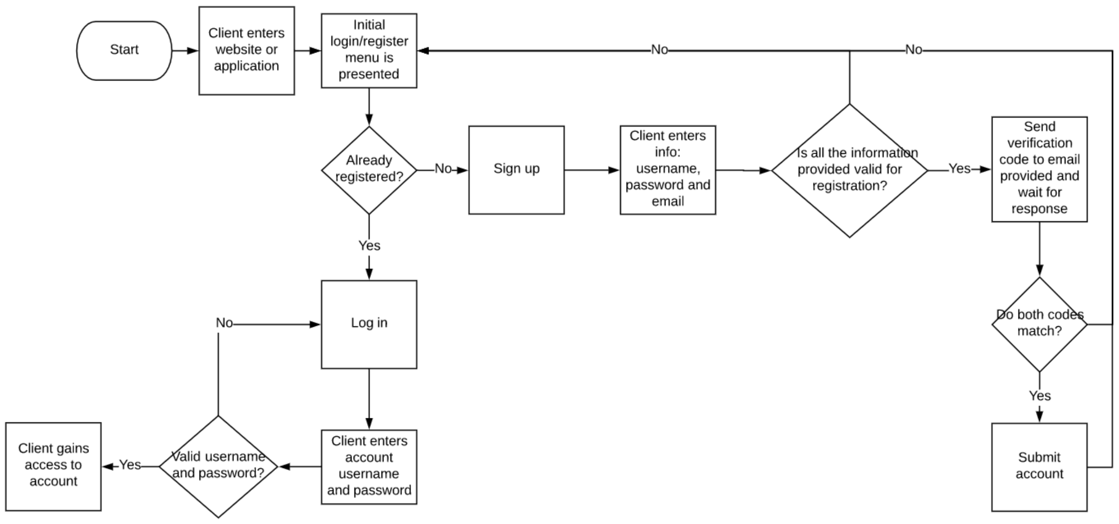

Start
Client enters website or application
Initial login/register menu is presented
No
No
Already registered?
No
Sign up
Client enters info: username, password and email
Is all the information provided valid for registration?
Yes
Send verification code to email provided and wait for response
Yes
Log in
No
Do both codes match?
Yes
Client gains access to account
Yes
Valid username and password?
Client enters account username and password
Submit account

Figure 2: Activity diagram for login and registration prototype

3 Prototype

The prototype has been implemented to show the registration and login processes in the perspective of a customer. No front end has been included as the main functionality can simply be shown from console.

The structural design consists of a client server, a web server and a database. In simple terms, the client server can input data into the console, and send it over to the server side. The server then reads this data and analyzes it before taking any action. Once the data has been checked, the server interacts with the database to perform the required operation. Finally, some form of response is sent back to client to let them know what to do next. (For more detailed explanation of functionality please refer to Server.java and Client.java files).

The prototype includes several security features which will be briefly mentioned in this section:

- The output stream files are always encrypted before sending from one server to another to render a "Man-in-the-middle" attack less effective. The encryption keys are stored in a keystore with a password. The opposite server can then receive the file and decrypt it using the same encryption key.
- The use of SQLite leaves vulnerabilities to SQL injections so these are prevented by the use of preparedStatements and also in some cases, sanitized input fields.
- The prototype also includes the use of two factor authentication through Gmail. During registration, the customer must enter an email and a one time code [6] is sent to it. The customer must then enter this code into the console and have it match with the original code in order to verify himself/herself.
- The passwords are always salted and hashed before storage to mitigate risk in case the data from the database is stolen. The "SHA1PRNG" algorithm and the "PBKDF2WithHmacSHA1" algorithm are used for salting and hashing respectively[1].
- The inputs are checked during registration for events such as already existing account under the same username or email, or password not complying with criteria (minimum of 8 characters, one upper case and lower case letter and one special character).
- Passwords are required to be inputted twice in case they were written incorrectly the first time by accident.

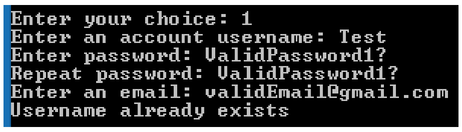
Table 1: Initial "Customers" table in the database for testing

CustomerID	Username	Password	Salt	Email
1	Test	Password_test	salt_test	Test@gmail.com

4 Evaluation

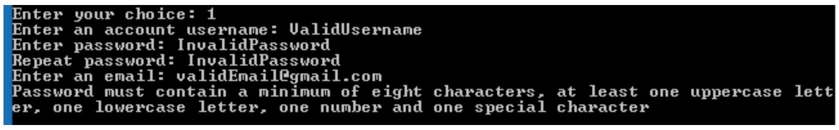
For easier understanding of the evaluations, here is the initial table of values within the database for every test.

Note: The console input concealer for the passwords was disabled for testing purposes whenever a password test was required.



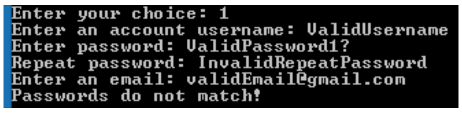
```
Enter your choice: 1
Enter an account username: Test
Enter password: ValidPassword1?
Repeat password: ValidPassword1?
Enter an email: validEmail@gmail.com
Username already exists
```

Figure 3: Username already exists



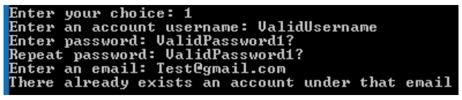
```
Enter your choice: 1
Enter an account username: ValidUsername
Enter password: InvalidPassword
Repeat password: InvalidPassword
Enter an email: validEmail@gmail.com
Password must contain a minimum of eight characters, at least one uppercase lett
er, one lowercase letter, one number and one special character
```

Figure 4: Password is not strong enough



```
Enter your choice: 1
Enter an account username: ValidUsername
Enter password: ValidPassword1?
Repeat password: InvalidRepeatPassword
Enter an email: validEmail@gmail.com
Passwords do not match!
```

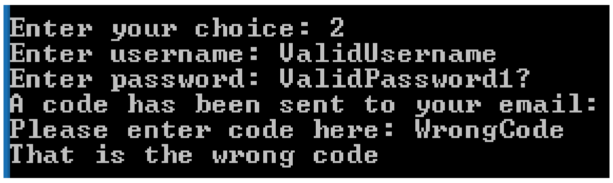
Figure 5: Password and repeated password do not match



```
Enter your choice: 1
Enter an account username: ValidUsername
Enter password: ValidPassword1?
Repeat password: ValidPassword1?
Enter an email: Test@gmail.com
There already exists an account under that email
```

Figure 6: Account already exists under that email

Figures 3 to 6 simply show the four different types of errors possible upon registration.

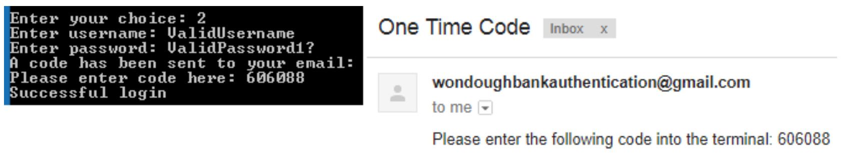


```
Enter your choice: 2
Enter username: ValidUsername
Enter password: ValidPassword1?
A code has been sent to your email:
Please enter code here: WrongCode
That is the wrong code
```

Figure 7: OTP's are not matching

Two-factor authentication allows the bank to confirm a user's identity and is only allowed to access his account if both the password and the code are correct as shown in figure 8.

Figure 9 shows the stored hashed password (the string underlined in red) and the salt (the string underlined in blue). As you can see, the password stored is completely different to the original password which helps mitigate the risk in the case the hashed password is obtained by an attacker.



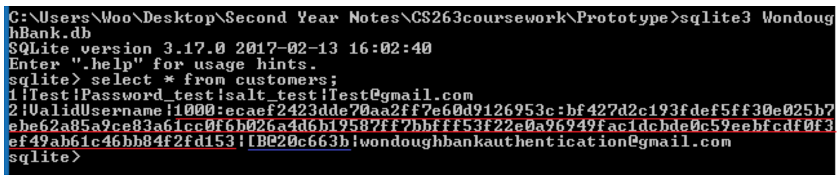
```
Enter your choice: 2
Enter username: ValidUsername
Enter password: ValidPassword1?
A code has been sent to your email:
Please enter code here: 606088
Successful login
```

One Time Code Inbox x

wondoughbankauthentication@gmail.com
to me

Please enter the following code into the terminal: 606088

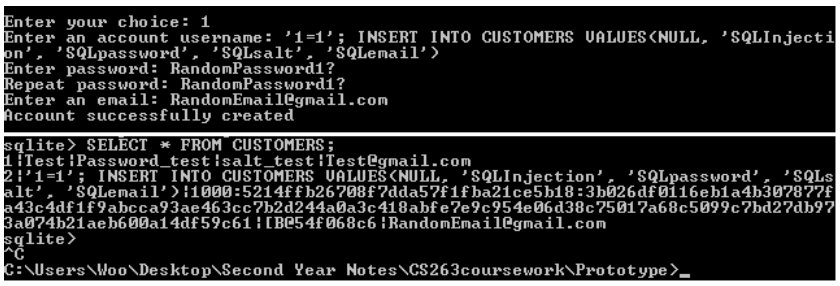
Figure 8: Email received with OTP and account created



```
C:\Users\Woo\Desktop\Second Year Notes\CS263coursework\Prototype>sqlite3 Wondoug
hBank.db
SQLite version 3.17.0 2017-02-13 16:02:40
Enter ".help" for usage hints.
sqlite> select * from customers;
1|Test|Password_test|salt_test|Test@gmail.com
2|ValidUsername|1000:ecaef2423dde70aa2ff7e60d9126953c:bf427d2c193fdef5ff30e025b7
ebe62a85a9ce83a61cc0f6b026a4d6b19587ff7bbfff53f22e0a96949fac1dcbde0c59eebfcdf0f3
ef49ab61c46bb84f2fd153|[B@20c663b|wondoughbankauthentication@gmail.com
sqlite>
```

Figure 9: Stored hashed password and salt

In figure 10, an SQL injection is attempted during registration to insert a new account into the database. However, thanks to the preparedStatements, the actual statement inputted in the console ends up being the username string and the account is created as normal.



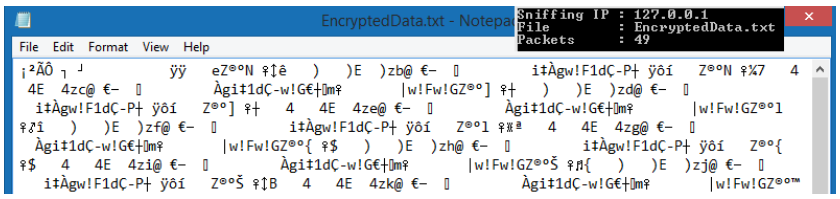
```
Enter your choice: 1
Enter an account username: '1=1'; INSERT INTO CUSTOMERS VALUES(NULL, 'SQLInjecti
on', 'SQLpassword', 'SQLsalt', 'SQLemail')
Enter password: RandomPassword1?
Repeat password: RandomPassword1?
Enter an email: RandomEmail@gmail.com
Account successfully created
```

```
sqlite> SELECT * FROM CUSTOMERS;
1|Test|Password_test|salt_test|Test@gmail.com
2|'1=1'; INSERT INTO CUSTOMERS VALUES(NULL, 'SQLInjection', 'SQLpassword', 'SQLs
alt', 'SQLemail')|1000:5214ffb26708f7dda57f1fba21ce5b18:3b026df0116eb1a4b307877f
a43c4df1f9abcca93ae463cc7b2d244a0a3c418abfe7e9c954e06d38c75017a68c5099c7bd27db97
3a074b21aeb600a14df59c61|[B@54f068c6|RandomEmail@gmail.com
sqlite>
^C
C:\Users\Woo\Desktop\Second Year Notes\CS263coursework\Prototype>_
```

Figure 10: SQL Injection has failed due to preparedStatements

The packages sent from the client to the web server have been caught and read into a file called "EncryptedData.txt" and figure 11 shows that the data that is been transmitted from one side to another is indeed completely encrypted and even in the case the attackers get their hands on such packages, they would require the encryption keys to decrypt the data and actually read the contents of the files. The program that allowed the capture and sniffing of packets over networks was [5] RawCap.

Brute force and combination attacks were also two of the tests that were possible to do however, due to the time constraints and the GPU capacity of the machine being currently used to run this prototype, the password cracker will take too long to actually de-hash the passwords stored in the database. The test could have been been completed by running the HashCat executable with the corresponding settings for each type of attack desired. HashCat also produces an estimation of the time required to crack the passwords and even when trying to crack five MD5 passwords, the program expects it to take approximately half a day. Hence, this test has been left uncompleted in expectation of the actual program taking too long to crack passwords that have been hashed under more challenging algorithms such as "PBKDF2WithHmacSHA1" (not even considering the salt).



```
Sniffing IP : 127.0.0.1
File        : EncryptedData.txt
Packets     : 49
```

```
¡²ÃÔ ┐ ┘          ÿÿ   eZ®°N ♀↨ê    )    )E  )zb@ €–  ▯         i‡Àgw!F1dÇ-P┼ ÿôí   Z®°N ♀¼7    4
 4E  4zc@ €–  ▯         Àgi‡1dÇ-w!G€┼▯m♀        |w!Fw!GZ®°] ♀┼    )    )E  )zd@ €–  ▯
  i‡Àgw!F1dÇ-P┼ ÿôí   Z®°] ♀┼    4    4E  4ze@ €–  ▯          Àgi‡1dÇ-w!G€┼▯m♀        |w!Fw!GZ®°1
♀♂î    )    )E  )zf@ €–  ▯          i‡Àgw!F1dÇ-P┼ ÿôí   Z®°1 ♀※ª   4    4E  4zg@ €–  ▯
  Àgi‡1dÇ-w!G€┼▯m♀        |w!Fw!GZ®°{ ♀$    )    )E  )zh@ €–  ▯          i‡Àgw!F1dÇ-P┼ ÿôí   Z®°{
♀$    4    4E  4zi@ €–  ▯          Àgi‡1dÇ-w!G€┼▯m♀        |w!Fw!GZ®°Š ♀♬{    )    )E  )zj@ €–  ▯
   i‡Àgw!F1dÇ-P┼ ÿôí   Z®°Š ♀↨B    4    4E  4zk@ €–  ▯          Àgi‡1dÇ-w!G€┼▯m♀        |w!Fw!GZ®°™
```

Figure 11: File with data being exchanged from server and client

References

[1] Server and Client https://www.pixelstech.net/article/1445603357-A-HTTPS-client-and-HTTPS-server-demo-in-Java

[2] Wordlist for HashCat http://www.md5this.com/tools/wordlists.html

[3] HashCat https://www.youtube.com/watch?v=U1MOUOEynyw

[4] DDOS prevention https://www.globalsign.com/en/blog/how-to-prevent-a-ddos-attack-on-a-cloud-serv

[5] RawCap http://www.netresec.com/?page=Blog&month=2011-04&post=RawCap-sniffer-for-Windows-released

[6] OTP Email Sender https://www.tutorialspoint.com/java/java_sending_email.htm

[7] Hidden Password Input https://www.tutorialspoint.com/java/io/console_readpassword.htm

[8] Regex Pattern https://stackoverflow.com/questions/19605150/regex-for-password-must-contain-at-least-eight-characters-at-least-one-number-a

[9] Hashing and Salting Passwords https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/

[10] Data Protection Act https://en.wikipedia.org/wiki/Data_Protection_Act_1998

[11] Financial Services and Markets Act 2000 https://en.wikipedia.org/wiki/Financial_Services_and_Markets_Act_2000