Credit card Fraud Detection

Final Project: EE 5453- Cloud Computing for Science and Engineering- Fall 2014

By: Abhishek Mukherjee (@01418070)

**CONTENTS:**

1. **ABSTRACT**
2. **METHOD USED**
3. **PROJECT DESCRIPTION**
4. **RESULTS**
5. **FUTURE MODIFICATIONS**
6. **CONCLUSION**
7. **REFERENCES**
8. **APPENDIX**

**Abstract**

*In recent years, financial institutions have emerged as the most attractive target of the Cyber Criminals. One such financial institution is gearing up to match the need of the hour and has contacted us to provide a solution to detect the possible anomalies that could be already present in their system. We have received the banking transaction details of few users and need to develop our solution to detect the possible compromise.*

**Introduction:**

Browsing through the reports of the cyber attacks in the past, the project which works from the vision of a solution provider comes across various patterns that Cyber Criminals follow depending on the present security of the target. To know the security measures in place, an attacker tries to reach that one or more than one host machine which is not updated with recent patches or Anti Virus software. Once the hacker plants the malicious code in that host machine, they begin attacking, stepping up from a small levels on the main database of the institution to bigger attacks.

**Concept:**

Based on the Observations, it was found that the hacker might choose to go for few small level of attacks before making a big move. At this point it is assumed that we (being the vendor), are already observant and knowledgeable about emerging trends in cyber crime. Extracting few cents out of few user bank accounts which can easily be neglected by the user when viewing the user copy of the monthly transaction report becomes a task which is not easily noticeable. We will read such patterns and inform the Bank about the occurrences, providing the exact date, location and frequency of transactions. Feeding the data received from the bank in SQlite3 database in Python and then using dictionaries to analyze and separate the targeted anomalies as and when the concerns arise. This is done so that this database can be monitored in real time and any concerns which seem fraudulent can be brought to the bank’s notice at the first instance.

**Project Description:**

This Solution is built using SQlite3 which is integrated in Python. The information received from the bank, has been split into different tables of SQLite. The idea is to have the database updated in real time and extraction of any fraudulent data which is identified based on certain guidelines in this case which is assumed to be three of them.

1. The first anomaly is identified when there is any discrepancy for amount withdrawals less than a certain amount from the user’s account which is unlikely to be have taken place.
2. Second Anomaly is based on Frequency. If there are transaction on the same date multiple times.
3. Third Anomaly is based on the Preferred location of an individual user which is gathered as per the trend provided.

Each user has been assigned an ID. This ID acts as a primary key/foreign key for other tables. . The following steps describe the working of the Solution in detail:

1. Five tables are created
2. ID
3. Preferred Location
4. Account
5. Amount
6. Frequency
7. To detect the Fraud using ‘Amount’ parameter, we have set a threshold value of 10$. Any transaction less than this value will be recorded. The date and User Account number for that transaction will be saved in an excel sheet which will be further used to draw the results.
8. If there are more than two transactions which are less than 10$, we have recorded that as well.
9. We received details from bank which has records of ‘Preferred location’ where a user can use the Credit card. In this Solution we have accepted Zip codes as a record from the bank.
10. The Dictionary function was used to create key value pair for Date and Amount. Dictionary function was used to determine the number of transaction that occurred for less than 10$ value.
11. These segregated values are then compiled on one data sheet as per the number of Users.

**Results:**

For all the results received from the steps above, an analysis is performed and can be put in a graphical representation shown in Figure 1:

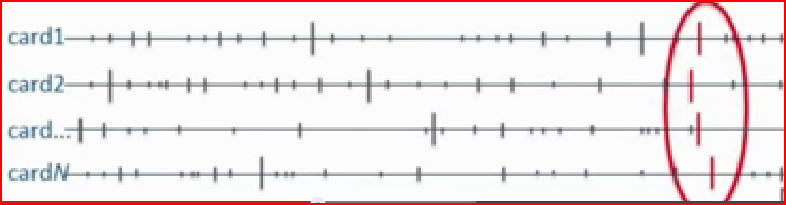


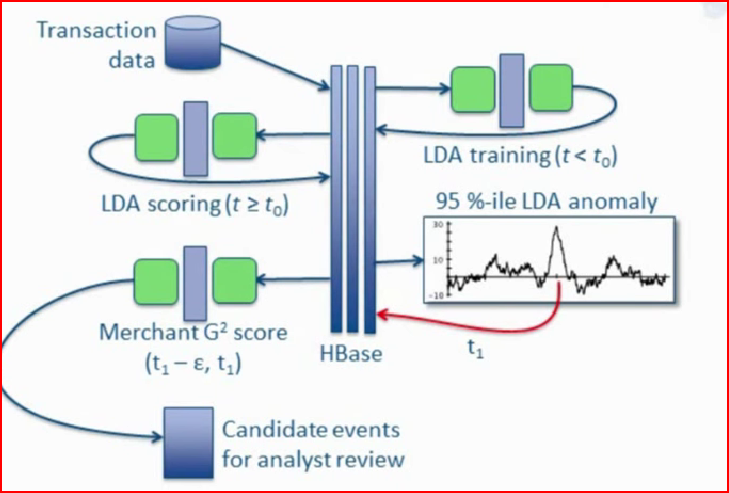
Fig.1

The x-axis in figure 1 is ‘Time and Date’ and Y-axis the ‘Card numbers’ used for the transaction. The small lines represents the transaction amounts during which are greater than 10$, however we have highlighted the transactions for less than 10$ using the taller lines because we are using that as a threshold for this Solution. As we can see from figure 1, that once the data provided by us is integrated into one single form, a pattern of attack could be detected. The area marked in the figure in red represents that: There has been more than 2 transactions, for less than 10$ at non-Preferred locations at the same time. There will be a very less probability that ‘n’ number of users are performing transaction at the same time, and all from the non-preferred location. This could be termed as a possible Compromise of their credit card. However, we leave the decision to the bank to use their ways of communication to reach the user to inform about the anomalies.

**Future Improvements:**

Just like any other software, this Solution is also open to few improvements which would be specific to the need of the Institution which implements it. Here are few of them

1. This solution could be integration with Android Technology which could facilitate a faster way of communicating the user or the Bank Officials. It could be further integrated with Exchange Services like Gmail or Outlook for automatic e-mails as soon as anomalies are detected.
2. Integrating the present Solution with Hadoop Database could meet the need of processing the data faster.



1. A thorough description of the Guidelines is required from the institution implementing the Solution. This would modify the data in the tables as per the detection parameter requested by the institution.

**Conclusion:**

The results shows that the emerging trends in Cyber Crime could be tamed using the Solutions like the one mentioned in this report. Also, such solutions could be modified further to integrate with Hadoop and Android Technology. As the knowledge on SQL and Python improves, the software part of the solution could be modified further to detect real-time fraud issues.

**References:**

1. <https://www.youtube.com/watch?v=Xt39pNZXVcg>
2. <http://www.volance.com/small_business.php>
3. <http://www.capterra.com/financial-fraud-detection-software/spotlight/89096/Arbutus%20Audit%20Analytics/Arbutus%20Software>

**Appendix**

The Python Code for the Solution

**import** sqlite3  
**import** datetime  
#import os  
**import** xlrd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**import** csv  
**import** xlsxwriter  
  
file\_location**="D:\Cloud Computing\Final\_Project\bank\_info.xlsx"**wbook**=** xlrd.open\_workbook**(**file\_location**)**conn **=** sqlite3.connect**('bank\_info.db')**c **=** conn.cursor**()**# Create table ID  
  
Sheetid**=**wbook.sheet\_by\_index**(**0**)**bank\_id **= []**c.execute**("DROP TABLE IF EXISTS id")**c.execute**('''CREATE TABLE id  
 (User Name,   
 ID,  
 CardNumber)''')  
  
for** rowx **in** xrange**(**Sheetid.nrows**):** bank\_id.append**(**tuple**(**Sheetid.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**Sheetid.ncols**)))**c.executemany**("INSERT INTO id VALUES (?,?,?)"**, bank\_id**)**conn.commit**()**# Create table preferred\_loc  
Sheetpl**=**wbook.sheet\_by\_index**(**1**)**bank\_pl **= []**c.execute**("DROP TABLE IF EXISTS prefloc")**c.execute**('''CREATE TABLE prefloc  
 (ID,  
 Preferred Location)''')  
  
for** rowx **in** xrange**(**Sheetpl.nrows**):** bank\_pl.append**(**tuple**(**Sheetpl.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**Sheetpl.ncols**)))**c.executemany**("INSERT INTO prefloc VALUES (?,?)"**, bank\_pl**)**conn.commit**()**# Create table account  
  
Sheetaccount**=**wbook.sheet\_by\_index**(**2**)**bank\_account **= []**c.execute**("DROP TABLE IF EXISTS account")**c.execute**('''CREATE TABLE account  
 (ID,  
 Account)''')  
  
for** rowx **in** xrange**(**Sheetaccount.nrows**):** bank\_account.append**(**tuple**(**Sheetaccount.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**Sheetaccount.ncols**)))**c.executemany**("INSERT INTO account VALUES (?,?)"**, bank\_account**)**conn.commit**()**# Table of Amount  
SheetAmount**=**wbook.sheet\_by\_index**(**3**)**bank\_Amount **= []**c.execute**("DROP TABLE IF EXISTS Amount")**c.execute**('''CREATE TABLE Amount  
 (ID,  
 amount,  
 Account Number)''')  
  
for** rowx **in** xrange**(**Sheetaccount.nrows**):** bank\_Amount.append**(**tuple**(**SheetAmount.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**SheetAmount.ncols**)))**c.executemany**("INSERT INTO Amount VALUES (?,?,?)"**, bank\_Amount**)**conn.commit**()**# Table of location  
  
Sheetloc**=**wbook.sheet\_by\_index**(**4**)**bank\_loc **= []**c.execute**("DROP TABLE IF EXISTS Location")**c.execute**('''CREATE TABLE Location  
 (ID,  
 location,  
 amount,Date)''')  
  
for** rowx **in** xrange**(**Sheetloc.nrows**):** bank\_loc.append**(**tuple**(**Sheetloc.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**Sheetloc.ncols**)))**c.executemany**("INSERT INTO Location VALUES (?,?,?,?)"**, bank\_loc**)**conn.commit**()**# Table of Frequency  
  
SheetFreq**=**wbook.sheet\_by\_index**(**5**)**bank\_Freq **= []**c.execute**("DROP TABLE IF EXISTS Frequency")**c.execute**('''CREATE TABLE Frequency  
 (ID,  
 Date,  
 amount)''')  
  
for** rowx **in** xrange**(**SheetFreq.nrows**):** bank\_Freq.append**(**tuple**(**SheetFreq.cell**(**rowx, colx**)**.value   
 **for** colx **in** xrange**(**SheetFreq.ncols**)))**c.executemany**("INSERT INTO Frequency VALUES (?,?,?)"**, bank\_Freq**)**conn.commit**()**#############################################  
# extracting all values for each id for non-preferred locations per user  
c.execute**('SELECT Location,amount,Date FROM Location WHERE ID==1 AND Location!=6845 AND Location!=1032 AND Location!=1975 AND Location!=6045 AND Location!=2950')**id\_1\_loc **=** c.fetchall**()**c.execute**('SELECT Location, amount,Date FROM Location WHERE ID==2 AND Location!=35345 AND Location!=59273 AND Location!=81813 AND Location!=75546 AND Location!=34076')**id\_2\_loc **=** c.fetchall**()**c.execute**('SELECT Location, amount,Date FROM Location WHERE ID==3 AND Location!=39372 AND Location!=39372 AND Location!=18658 AND Location!=46643 AND Location!=75561')**id\_3\_loc **=** c.fetchall**()**c.execute**('SELECT Location, amount,Date FROM Location WHERE ID==4 AND Location!=10111 AND Location!=19907 AND Location!=46082 AND Location!=59521 AND Location!=59524')**id\_4\_loc **=** c.fetchall**()**# extracting all values for each id for amounts less than 10$ and their respective dates  
c.execute**('SELECT amount,Date FROM Frequency WHERE ID==1 AND amount<=10')**id\_1\_amt **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==2 AND amount<=10')**id\_2\_amt **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==3 AND amount<=10')**id\_3\_amt **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==4 AND amount<=10')**id\_4\_amt **=** c.fetchall**()**# extracting all values for each id for consecutive dates  
c.execute**('SELECT amount,Date FROM Frequency WHERE ID==1')**id\_1 **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==2')**id\_2 **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==3')**id\_3 **=** c.fetchall**()**c.execute**('SELECT amount,Date FROM Frequency WHERE ID==4')**id\_4 **=** c.fetchall**()**##############  
#create dictionary for Date and amount(1 date, two related amount)  
**def dict\_amt\_date(**id\_0**):** l **=** len**(**id\_0**)** id\_0\_fr**={}  
 for** row **in** range **(**0,l**-**1**):** old\_date **=** id\_0**[**row**][**1**]** new\_date **=** id\_0**[**row**+**1**][**1**]  
  
 if (**old\_date **==** new\_date**):** v1**=**id\_0**[**row**][**0**]** v2 **=** id\_0**[**row**+**1**][**0**]** k **=** old\_date  
 id\_0\_fr**[**k**] =** v1,v2  
   
 **return (**id\_0\_fr**)**id\_1\_fr **=** dict\_amt\_date**(**id\_1**)**id\_2\_fr **=**dict\_amt\_date**(**id\_2**)**id\_3\_fr **=**dict\_amt\_date**(**id\_3**)**id\_4\_fr **=**dict\_amt\_date**(**id\_4**)**#########################################3  
  
#create dictionary for Date and amount<$10(1 date, 1 amount)  
**def dict\_lessamt\_date(**id\_0**):** l **=** len**(**id\_0**)** id\_0Dateamt**={}  
 for** row **in** id\_0**:** v1**=**row**[**0**]** #amount  
 k **=** row**[**1**]** # date  
 id\_0Dateamt**[**k**] =** v1  
   
 **return (**id\_0Dateamt**)**id\_1\_lessamt **=** dict\_lessamt\_date**(**id\_1\_amt**)**id\_2\_lessamt **=**dict\_lessamt\_date**(**id\_2\_amt**)**id\_3\_lessamt **=**dict\_lessamt\_date**(**id\_3\_amt**)**id\_4\_lessamt **=**dict\_lessamt\_date**(**id\_4\_amt**)**#################################################  
#create dictionary for non-preferred locations  
**def dict\_nonPre\_loc(**id\_0**):** l **=** len**(**id\_0**)** id\_nonPreLoc**={}  
 for** row **in** id\_0**:** v1**=**row**[**0**]** # location  
 k **=** row**[**1**]** # amount  
 id\_nonPreLoc**[**v1**] =** k  
   
 **return (**id\_nonPreLoc**)**id\_1\_nonPreLoc **=** dict\_nonPre\_loc**(**id\_1\_loc**)**id\_2\_nonPreLoc **=**dict\_nonPre\_loc**(**id\_2\_loc**)**id\_3\_nonPreLoc **=**dict\_nonPre\_loc**(**id\_3\_loc**)**id\_4\_nonPreLoc **=**dict\_nonPre\_loc**(**id\_4\_loc**)**User1**={}**User1**={'LessAmount':[**id\_1\_lessamt**]**, **'FreqDate':[**id\_1\_fr**]**, **'NonPrefLocation':[**id\_1\_nonPreLoc**] }**User2**={}**User2**={'LessAmount':[**id\_2\_lessamt**]**, **'FreqDate':[**id\_2\_fr**]**, **'NonPrefLocation':[**id\_2\_nonPreLoc**] }**User3**={}**User3**={'LessAmount':[**id\_3\_lessamt**]**, **'FreqDate':[**id\_3\_fr**]**, **'NonPrefLocation':[**id\_3\_nonPreLoc**] }**User4**={}**User4**={'LessAmount':[**id\_4\_lessamt**]**, **'FreqDate':[**id\_4\_fr**]**, **'NonPrefLocation':[**id\_4\_nonPreLoc**] }**###################################################################################################  
  
  
**def createExeclFile(**name, id\_0\_loc,id\_0, id\_0\_amt**):** workbook **=** xlsxwriter.Workbook**(**name**)** worksheet **=** workbook.add\_worksheet**('NonPrefLocation')** worksheet1 **=** workbook.add\_worksheet**('FreqDate')** worksheet2 **=** workbook.add\_worksheet**('LessAmount')** row **=** 0  
 col **=** 0  
 **for** a,b,c **in (**id\_0\_loc**):** worksheet.write**(**row, col, a**)** worksheet.write**(**row, col **+** 1, b**)** worksheet.write**(**row, col **+** 2, c**)** row **+=** 1  
 row **=** 0  
 col **=** 0  
 **for** a,b **in (**id\_0**):** worksheet1.write**(**row, col, a**)** worksheet1.write**(**row, col **+** 1, b**)** row **+=** 1  
 row **=** 0  
 col **=** 0  
 **for** a,b **in (**id\_0\_amt**):** worksheet2.write**(**row, col, a**)** worksheet2.write**(**row, col **+** 1, b**)** row **+=** 1  
  
 workbook.close**()**createExeclFile**('USER1.xlsx'**, id\_1\_loc,id\_1, id\_1\_amt**)**createExeclFile**('USER2.xlsx'**, id\_2\_loc,id\_2, id\_2\_amt**)**createExeclFile**('USER3.xlsx'**, id\_3\_loc,id\_3, id\_3\_amt**)**createExeclFile**('USER4.xlsx'**, id\_4\_loc,id\_4, id\_4\_amt**)**

Data Output in Excel sheet which is further provided to the Bank as a Result:

|  |  |  |
| --- | --- | --- |
| Zip Code | Amount | Date |
| 31652 | 352 | 02/15/2014 |
| 3049 | 230 | 04/09/2014 |
| 532360 | 388 | 06/08/2014 |
| 51312 | 55 | 11/16/2014 |
| 3667 | 395 | 11/22/2014 |
| 69935 | 383 | 12/01/2014 |
| 8993 | 236 | 12/31/2014 |
| 6663 | 68 | 02/06/2015 |
| 7689 | 138 | 02/16/2015 |
| 64217 | 2 | 02/25/2015 |
| 17369 | 119 | 04/03/2015 |
| 30110 | 207 | 04/20/2015 |
| 22234 | 8 | 04/30/2015 |
| 4387 | 320 | 05/06/2015 |
| 206282 | 357 | 06/09/2015 |
| 4873 | 171 | 06/28/2015 |
| 6375 | 88 | 08/04/2015 |
| 55604 | 442 | 08/12/2015 |
|  |  |  |

Data output in form of Frequency of transaction on a particular date and the amount:

|  |  |
| --- | --- |
| 351 | 12/27/2013 |
| 107 | 12/29/2013 |
| 180 | 12/29/2013 |
| 219 | 01/05/2014 |
| 352 | 01/19/2014 |
| 464 | 02/08/2014 |
| 352 | 02/15/2014 |
| 199 | 02/17/2014 |
| 210 | 02/19/2014 |
| 86 | 03/04/2014 |
| 49 | 03/05/2014 |
| 135 | 03/08/2014 |
| 125 | 04/08/2014 |
| 230 | 04/09/2014 |
| 19 | 04/11/2014 |
| 303 | 04/13/2014 |
| 187 | 04/13/2014 |
| 20 | 04/24/2014 |
| 427 | 05/05/2014 |
| 240 | 05/09/2014 |
| 259 | 05/13/2014 |

Data output number of transactions below 10$ and the date the transaction has occurred:

|  |  |
| --- | --- |
| 5 | 09/05/2014 |
| 2 | 02/25/2015 |
| 8 | 04/30/2015 |