*Athabasca University*

# Stopping Account Fraud

Intrusion and Fraud Detection with Artificial Intelligence

Philip Kirkbride
3060988
COMP 601
Richard Huntrods
October 20, 2019

## Intrusion and Fraud Detection with Artificial Intelligence

In the last decade simple cyber attacks have grown in popularity. In my first paper, *Credential Stuffing for Script Kiddies and Career Criminals*, I discussed how software like OpenBullet is making effective methods of cracking accounts on large websites more accessible to even amateur hackers or vetran criminals with little IT experience. As this software grows more widespread our way of managing accounts and identities online is threatened.

## Limits of 2FA

Some solutions were outlined for example the user of 2FA (two-factor authentication), but if the root email account used for 2FA is compromised then there is no defence left. Phone numbers can also be used for 2FA but in recent years there have been several cases of "SIM swapping" in which a malicious actor commits identity fraud with a phone company in order to move the associated account to a SIM card controlled by the attacker. Once the SIM card is obtained all 2FA which rely on the phone number can be compromised.

In the cases reported by the NY Post the assets stolen were worth more than $23 million. Due to the resource intensivity of this kind of attack it is mainly reserved for high profile victims. Yet this attack demonstrates the weakness of 2FA. As with captcha, 2FA is a great deterrent but not sufficient protection in many cases.

## Artificial Intelligence for Intrusion and Fraud Detection

Of course no solution will stop attacks completely but the next evolution in protection seems to revolve around the use of AI for fraud detection and account break ins. This is where the aggregate of all user interactions on a server are compiled into a big data set. From this big data set cases of identity fraud are identified and used to train an artificial intelligence to detect fraudulent logins based on server interactions [2].

Once a model for detecting fraud has been created a filter is placed between a user's request and the final execution of that request. For example a user logs into a bank account, looks at balances and transactions, and finally makes a request to send an e-transfer. During this process several factors are being observed by the server and fed into the fraud detection model. This could include things like:

- The location a user is logging in from
- The speed at which the users clicks on links or types
- The behaviour of the user as compared to historical behaviour

The request for sending an e-transfer is considered to be of a higher security risk than simply logging in, at this point all the behaviour is evaluated against the model and a likelihood of fraud score is generated. Based on the fraud score a number of different actions can be taken by bank including:

- Putting an extra hold on the transaction (1 hour to multiple days)
- Requiring an additional step such as 2FA via email, or having the user call a live agent at the bank to verify that it is indeed them
- Locking the account and sending out alerts to the customer via multiple channels (email, phone, and mail)

## Supervised and Unsupervised Learning for Fraud and Intrusion Detection

Machine learning is generally categorized in three broad categories supervised, unsupervised, and reinforcement learning. Supervised learning may seem like the most obvious solution to fraud at first. This strategy would be undertaken by taking a large dataset of user interactions, and labelling them as “legitimate” or “fraudulent”. Then training a machine learning algorithm to identify transactions or user logins which resemble past fraudulent transactions.

The supervised method of fraud detection ends up having two main weaknesses. Firstly the amount of data of cases of fraudulent interactions on specific systems is low. Secondly because supervised learning trains of examples of past examples it is only trained to find occurrences which look like previously used methods [3]. Whenever new methods are created the model has to be updated to identify these patterns which may look different from those previously used.

Unsupervised learning on the other hand can be trained to simply find transactions or users which look like anomalies. An unsupervised learning model can pickup on completely new methods of fraud which appear different from the methods used in the past but are still out of the ordinary from normal user or site use [3].

## Anomalies at the User and System Levels

We should also mention that there are two major levels to consider anomalies at. There are anomalies at a system level and anomalies at a user level. For example if a user has a year’s history of logging in every two weeks and sending $100 worth of e-transfers and suddenly that user logs in and sends a $2000 e-transfer. This would be considered a local anomaly as it is unusual for the user, but it wouldn’t be considered unusual behaviour in general to send a $2000 e-transfer [3].

If on the other hand a user logged in and sent 50 transactions of $10 each in a short period of time it might be considered a system level anomaly, as this behaviour would be considered strange across the system. Though it is still possible there is a perfectly legitimate reason for the behaviour, and if done consistently by a user it may be considered not an anomaly on the user level. These two levels should both be considered when generating a fraud probability score.

One difficulty faced when generating a base-line of normality at the user-level is the lack of data for each user, as compared to the amount needed to properly train an unsupervised model for anomaly detection. Even after having a bank account for 6 months a user’s transaction and use history may not be sufficient to create a baseline of behaviour that won't trigger false positives.

An efficient solution to this problem is to use a nearest neighbour algorithm like kNN (k-nearest neighbours algorithm) [3] to transfer learning between similar users. A user with a relatively short history with the bank could be represented as an array of data including things like:

- Monthly income
- Location
- Where they shop
- Who they send money to
- Who sends them money

This data is then fed to the kNN algorithm which can return a list of the most similar users. The baseline of normality for the bank account is then generated by creating a hybrid of the user and older accounts which appear most similar to them.

**Mapping Algorithm Probability to System Security Actions**

As mentioned earlier at the system level we'll often want to trigger different system security actions in response to anomalies. The anomaly detection algorithm described earlier will generally take a user object and a transaction object and process it to return a fraud probability score:

```
user = {
  name: “john doe”,
  history: {...},
  ...
}

transaction = {
  time: “23:44”,
  amount: 500,
  ...
}

let fraudScore = getFraudScore(user, transaction);
```

Once the fraud score is obtained the system architect must consider several aspects the most important are:

- Preventing fraud
- User satisfaction
- Company resources

The first aspect should be relatively obvious so we won't go into detail.

User satisfaction, the system needs to be easy enough to use that users feel satisfied with the bank and its online services. If a user is forced to call the bank every time they try to send an e-transfer they'll soon stop using the bank's online services. Where as getting a 2FA code from an automated phone call to complete the transaction may be more acceptable to the user.

The third consideration is company resources, and this is particularly in relation to the resources consumed by any specific security action. Even a 2FA code sent by email or text message has some associated cost with it albeit a fraction of a cent. Requiring a user to call customer service to verify their identity consumes a paid employee's time creating an associated cost. Or a user action may trigger additional processing of a user to analyze the account in depth, this in turns costs power and server computing power, both limited resources.

Generally banks are willing to pay a high cost in resources in order to provide a good balance of user satisfaction and fraud prevention. But if we apply the same principles to an alternative system that might not be the case. A pizza company offering users points for free pizzas for example may consider phone based verification too expensive even when there is a high probability of fraud.

A simplified version of a fraud score to security action mapping might look like:

```
switch(true) {
  case (fraudScore > 0.9):
    // Lock account, alert other companies
  case (fraudScore > 0.75):
    // Require user to visit local bank
  case (fraudScore > 0.50):
    // Require user to call by phone and verify information
  Case (fraudScore > 0.2):
    // Send a 2FA message by email or text
}
```

Once such a system is in place the trigger values and actions can be tweaked to optimize the system for fraud prevention, user experience, and resource saving.

**Cross-Company and Industry Cooperation**

These strategies can be strengthened by having multiple companies work together. For example in the case that a bank account is locked due to a high chance of fraudulent login, additional companies may be altered (credit cards company, phone carrier, ect).

Cross-company cooperation will be more important than ever as this type of fraud detection will be resource intensive. While detection is certainly feasible for banks and companies like

Google a small e-store for example would not have the resources required to setup such precautions. In response several larger companies will encourage sign-on-as-a-service with both free versions like that offered by Google and other social media sites, and paid versions such as AWS Cognito offered by Amazon.

By lowering the amount of gateways used for authentication the cost to implementing fraud detection using AI will be lowered and security for the whole economy improved. Small companies gain by improving their own security and improving ease of use for end-users. Large companies providing these services also gain. In the example of AWS the benefit is a direct monetary one whereas with Google the benefit is being able to gain additional data and insights about their users.

There may also be benefits from cooperating through data sharing. For example a cluster of phone numbers, emails, or IP addresses used in fraud can be shared across companies to improve their data coverage. A scammer moving to a second bank after successfully frauding a first may be blocked off due to shared blacklists and data sharing [2].

## Citations

1. M. Kaplan, “Hackers are stealing millions in Bitcoin - and living like big shots,” *New York Post*, 15-Apr-2019. [Online]. Available: https://nypost.com/2019/04/13/hackers-are-stealing-millions-in-bitcoin-and-living-like-big-shots/. [Accessed: 20-Oct-2019].
2. C. Phua, V. Lee, K. Smith, and R. Gayler, “A Comprehensive Survey of Data Mining-based Fraud Detection Research,” *2010 International Conference on Intelligent Computation Technology and Automation*, 2010.
3. R. J. Bolton and D. J. Hand, “Unsupervised Profiling Methods for Fraud Detection ,” *Proc. Credit Scoring and Credit Control VII*, 2001.