Lifecycle of a Phone Fraudster: Exposing Fraud Activity from Account Reconnaissance to Takeover using Graph Analysis and Acoustical Anomalies

Abstract

Enterprises are vulnerable to "human hacking" the effective social engineering of employees, contractors and other trusted persons. In particular, financial institutions have seen a significant increase in account takeover attacks over the phone by sophisticated fraudsters socially engineering call center agents. The customer information required is often obtained by gathering intelligence through reconnaissance, probing systems or humans. In this talk we will show how to detect both the account takeover calls using acoustical anomalies and the reconnaissance calls leading to it through graph analysis. Using acoustical anomalies we are able to detect over 80% of these calls with less than a 2% false positive rate. Furthermore our graph analysis is able to see reconnaissance calls for 46% of these account takeovers 10 days before the actual takeover. These results are on a dataset of over hundreds of million calls. In the process we will reveal the lifecycle of a phone fraudster as he works through both the call center agent and its technology to extract information about a customer and takeover his or her account.

Introduction

Calling any enterprise typically involves first dealing with an interactive voice response (IVR) system which is self service and can be used for basic or low risk transactions (eg. account balance check) followed by dealing with an agent for more complicated or higher risk transactions (eg. wire transfer). In taking over an account over the phone we find that fraudsters both social engineer the call center agents and exploit the information revealed at the IVR.

Detecting social engineering using acoustical anomalies

The phone device making a telephone call and the network path that a call takes leave behind tell tale audio artifacts that reveal the device and the path. An example of an artifact is when a call goes over a VoIP network it experiences packet loss and that packet loss results in tiny breaks (order of 10 ms) in the call audio. Currently, we have 147 such audio features that have allowed us to create a phone print that uniquely identifies a device as well as identifies the kind of device and the geography that the call is coming from. In this talk we will detail example artifacts and call audio that illustrates how we identify brand new fraudsters based on audio anomalies, and then catch repeat fraudsters using phoneprinting. We will walk through results and real world case studies where this technology has achieved over 80% detection rate in identifying account takeover phone calls with a low false positive rate lesser than 2%.

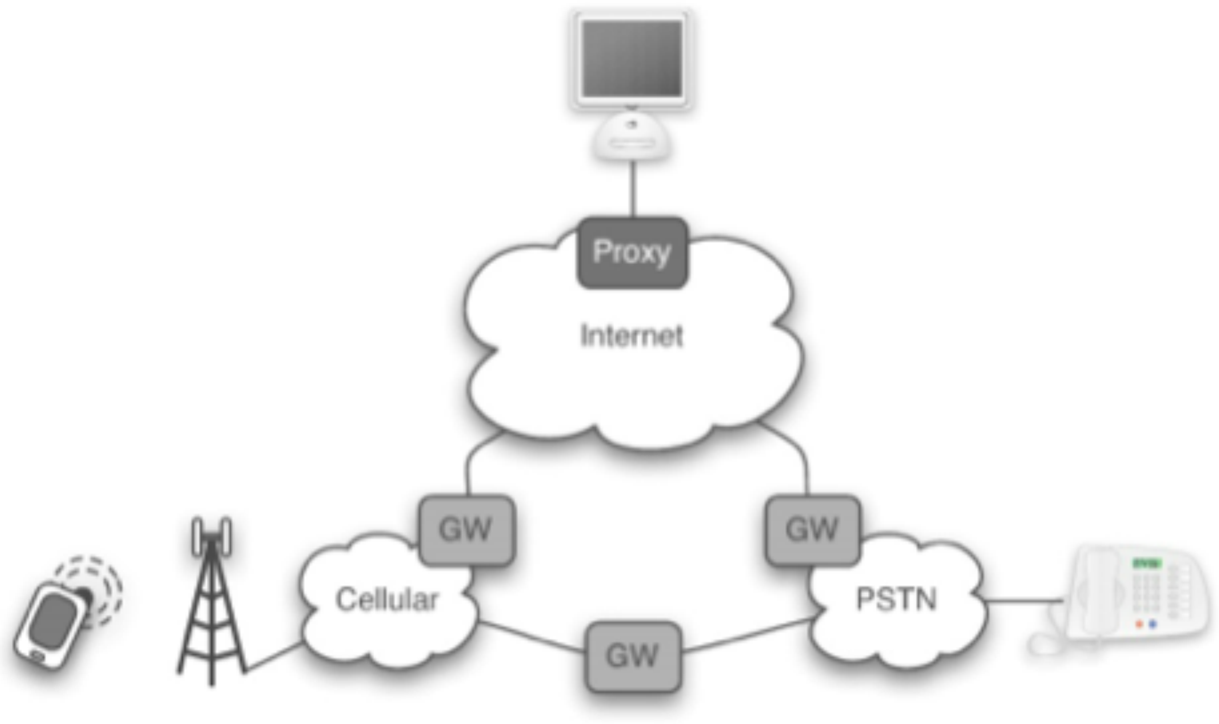
Detecting reconnaissance using graph analysis of Call Detail Record (CDR)

Fraudsters frequently exploit IVR systems leaking information by trying different menu options. For example, we see fraudsters run through a set of SSNs in the IVR to determine which ones are valid. We will demonstrate how we used graph analysis to look at CDRs and identify this and other reconnaissance activity. We show that for 46% of the account takeover calls the reconnaissance activity can be detected a whole 10 days before.

Both our acoustical anomaly and graph analysis detection mechanisms have currently been running over hundreds of millions of calls. They are completely passive and difficult for fraudsters to detect or evade. Finally, through this we will reveal the lifecycle of a phone fraudster as he works through both the call center agent and its technology to extract information about a customer and takeover his or her account.

Background

Telecommunication networks are exceedingly complex systems. While once designed, manufactured and run by a single company, today's networks are an elaborate combination of many different technologies. We offer a very high-level description of these systems including the signaling used in these systems and how voice is encoded in them.



As shown in the above figure, there are three general classes of telephony networks. PSTNs represent traditional circuit-switched telephony systems. These networks are generally characterized by lossless connections and high fidelity audio. While pieces of the core of some of these networks are being replaced by IP connections, these private links are tightly controlled to ensure near zero packet loss. Like PSTN systems, cellular networks have a circuit switched core, with portions currently being replaced by IP links. Finally, VoIP networks run on top of IP links and generally share the same paths as all other Internet-based traffic, and nearly always experience packet loss.

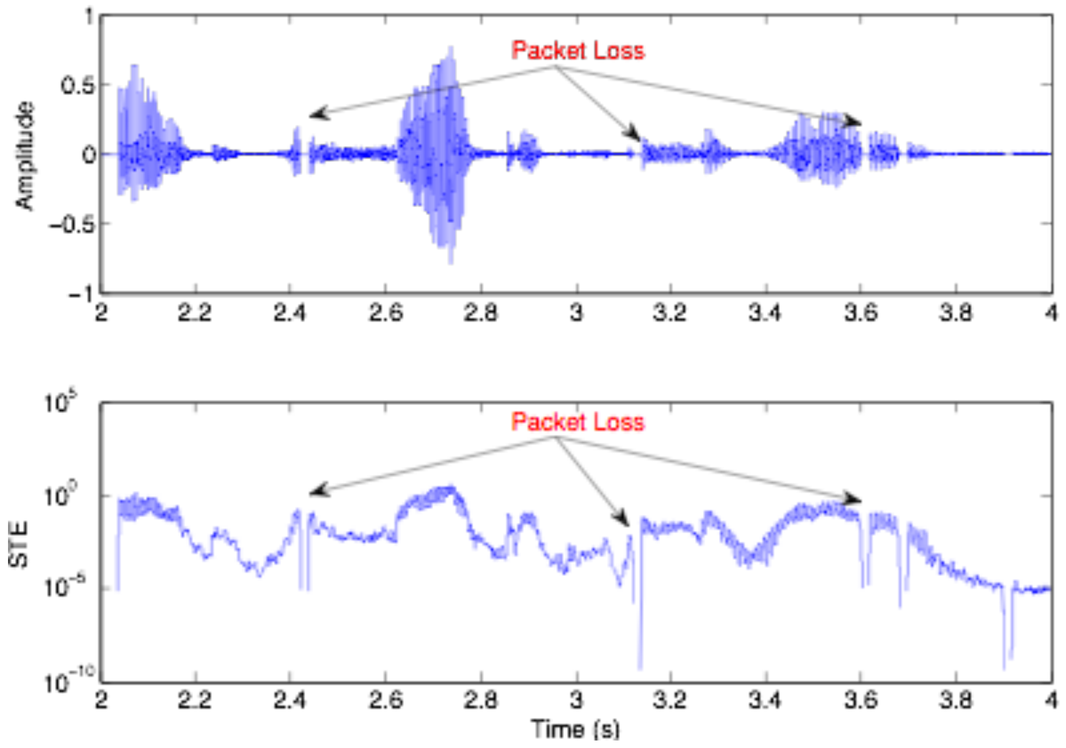
In all these networks there are two parts to enable calling, (a) signaling that establishes and tears down the call, and (b) media which carries the voices of the call participants. These are achieved by different mechanisms in each of these networks. The core signaling mechanism used for call setup, routing and control in PSTN and cellular networks is the common channel signaling system no. 7, SS7. The signaling mechanism for VoIP is similar to SS7 and is enabled using either the Session Initiation Protocol (SIP) proposed by the IETF, H.323 proposed by the ITU or proprietary protocols such as those used by Skype. When calls move from one type of network to another, the signaling is invariably discarded. With respect to media, voice is encoded and decoded in each of these networks using a variety of codecs. Specific codecs are selected for different network based on competing goals including sound quality, robustness to noise and bandwidth requirements. While a large number of codecs exist, the five most commonly used narrow band codecs and their typical environments are summarized in the table. When calls move from one type of network to another, the media is re-encoded to the new codec.

Detecting Social Engineering

We have created a Caller ID/ANI alternative by determining the provenance of a call - the source and the path taken by a call. Our technology is based on our research, PinDr0p, which shows that regardless of the claimed source, the audio delivered to a call recipient exhibits measurable features of the phone that made the call and the networks that were traversed by a call. We extract 147 features to develop detailed profiles or 'phoneprints' for various call sources. Here we explain just two of those features that we use to profile networks that a call has traversed: (i) VoIP, (ii) PSTN.

Profiling VoIP Networks Using Packet Loss

VoIP networks exhibit packet losses that are not seen in circuit switched PSTN or cellular networks. Accordingly, if we identify artifacts of these lost packets in the received audio we can deduce that the call traversed a VoIP network. The top graph in the figure below shows two seconds of speech transmitted through a VoIP network with a packet loss rate of 5%.



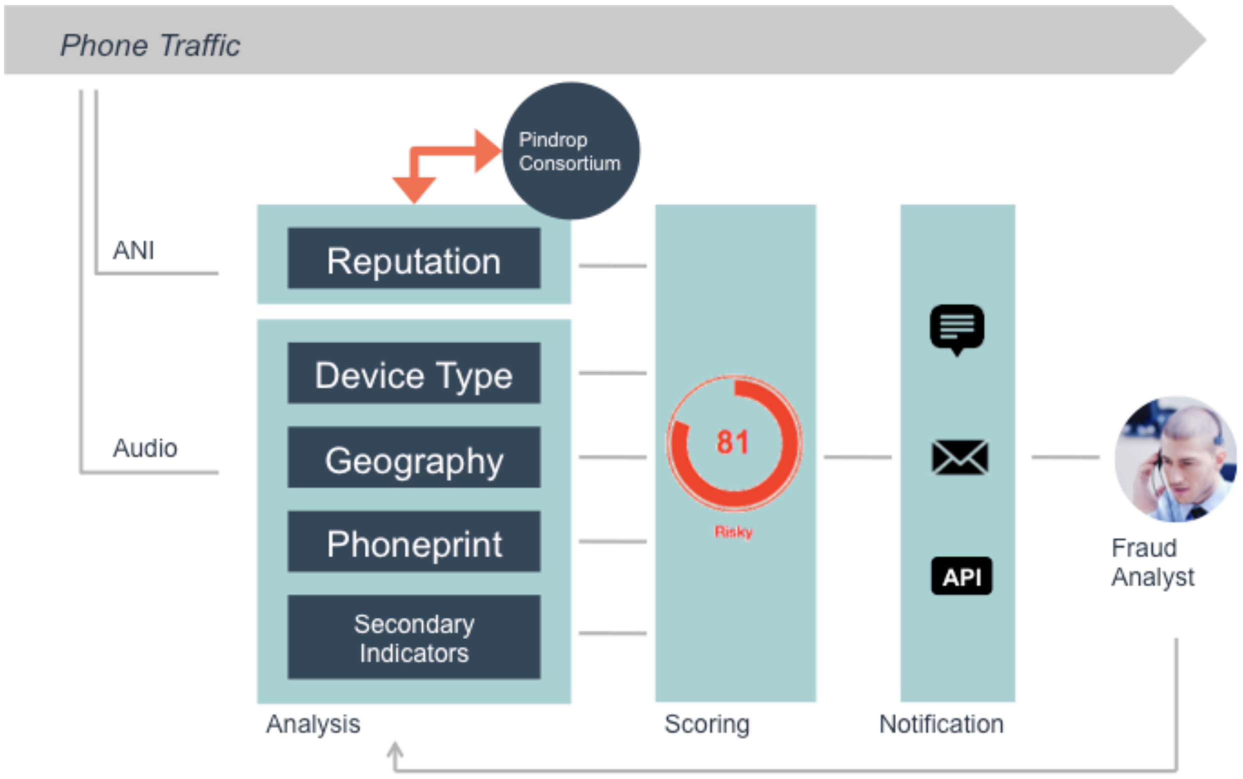
The effect of a lost packet is sometimes visibly identifiable by a break in the waveform (annotated by arrows). However, such loss can be detected more accurately by determining the short-time average energy of the signal, as shown in the bottom graph in the above figure. We detect packet loss by looking for a significant drop in energy followed by an energy floor, accompanied by an energy rise. Identifying all the packet losses provides a packet loss profile for the network that the call traversed. However, packet loss also reveals other parameters used within that network. To illustrate, each time a packet loss is detected, the length of the energy floor also reveals the codec used as iLBC encodes 30ms, Speex encodes 20 ms, G.729 encodes 10 ms and G.711 encodes 20 ms of speech per packet. We have found that packet loss alone is a rich source to determine other parameters set in that VoIP network and there are many other features that we use to profile a specific VoIP network.

Profiling PSTN Using Spectral Statistics

All PSTN systems use G.711, which by virtue of being a waveform codec, introduces, noise only during speech activity. This results in a strong correlation between the noise and the signal and its presence can be determined based on spectral statistic metrics: spectral level range and the spectral level deviation. In addition the values for spectral level range and spectral level deviation are two of the many features that go in building a noise profile for each call.

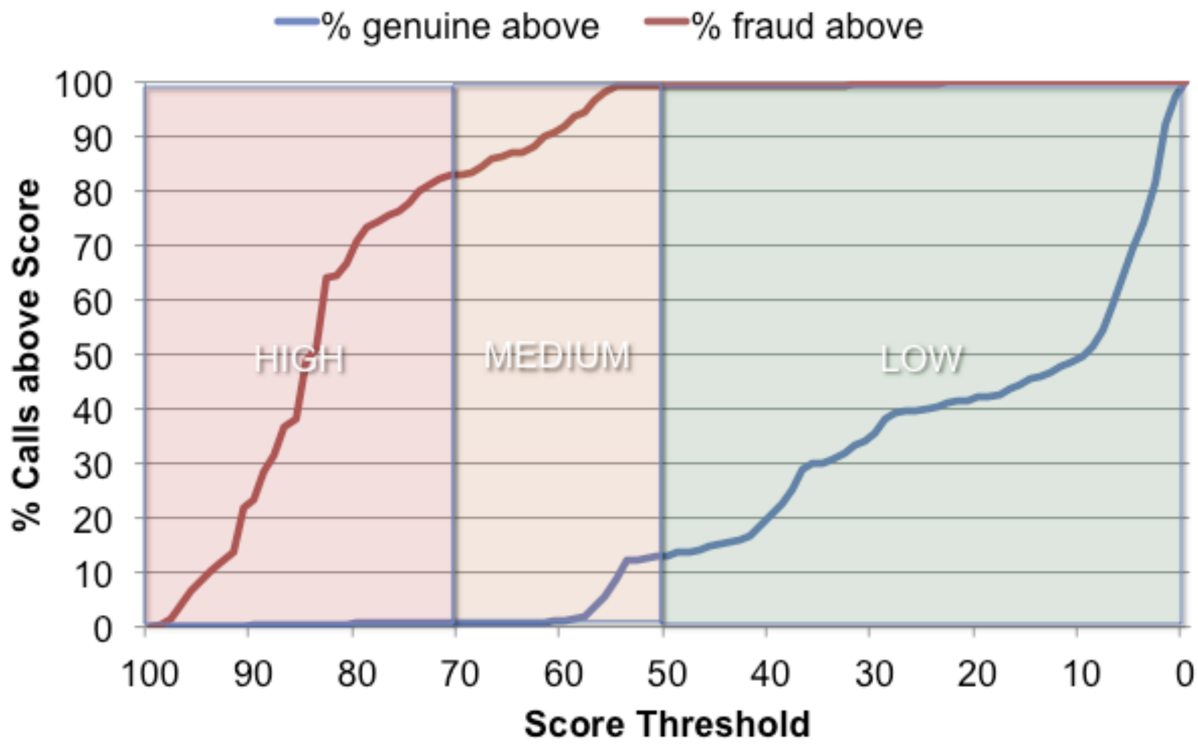
From Features to Detecting Fraud

The 147 features allow us to create machine learning models to identify the true type of device used to make a call, the geography a call is coming from, and finally identify the device itself through its phoneprint as shown in the figure below



Pindrop is able to separate legitimate and fraudulent traffic by using 4 primary engines:
1) Phone number reputation (ANI based) – this determines if an ANI has been reported doing bad activity by the consortium, is unallocated, is from risk carriers and a host of other reasons
2) Device Type (Audio based) – this determine if the call is actually coming from a risky VoIP system (eg Tuitalk). This is more risky if the caller is trying to use an ANI that is known to be a land or a cell and is an indicator for spoofing
3) Geography (Audio based) – this determines if the call is actually coming from a risky international geography (eg Nigeria). This is more risky if the caller is trying to use an ANI that is known to be domestic and is an indicator for spoofing
4) Phoneprint (Audio based) – this determines if the call is coming from a well known fraudster phone device. Our phoneprint is able to identify if it is the same device even if the fraudster changes/spoofs the ANI, hides behind VoIP gateways or changes/distorts his voice
Anomalies in 2) and 3) identify brand new fraudsters while 1) and 4) identify repeat fraudsters

Each of the engines creates a risk factor. We then use active learning to determine the appropriate risk weights for the total set of risk factors. Finally these risk factors and their risk weights are combined probabilistically to create a risk score between 0 and 100. Based on a subset of risk factors and a threshold on the resulting risk score we can decide what calls will be marked suspicious. The threshold determines an operating point that allows for a certain fraud detection rate while maintaining a certain false positive rate. Over the dataset the figure below shows the final lift curve for both legitimate calls and fraudulent calls.



This graph shows that you can pick a set of operating points to bucket your customer calls into low, medium and high risk. This allows you to determine a treatment that

a) Stops fraud
b) Reduces call center time
c) Improves customer experience

For example you can set all calls below 50 as low risk as only 0.8% (lesser than a percent) of fraud calls score below 50. A medium risk bucket can be between 50 and 80 and a high-risk bucket can be above 80 above which only 0.6% of your genuine calls lie.

Detecting Account Reconnaissance

Audio analysis is an extremely effective method of detecting fraud and spoofing. However, phone-based systems collect other kinds of useful metadata, typically called Call Detail Records (CDR). By analyzing large volumes of CDR data from a phone system, we can discern behavioral patterns that can often clearly identify fraudulent activity.

CDR analysis for detecting fraudulent activity usually involves the following steps:

1. Design features from CDR data to represent each call as a numeric feature vector.
2. Use a labeled set of feature vectors to train a machine learning model
3. Use the trained model for scoring the riskiness of each call.

Characteristics of CDR data

A typical CDR data set has at least one record for each phone call, typically produced at the end of a call. The exact schema of the CDR records vary widely depending on the domain, however, most of them contain at least the following pieces of information:

- Start and end timestamps of the call
- The originating ANI of the call (as shown by the caller ID)
- If applicable, the destination ANI of the call

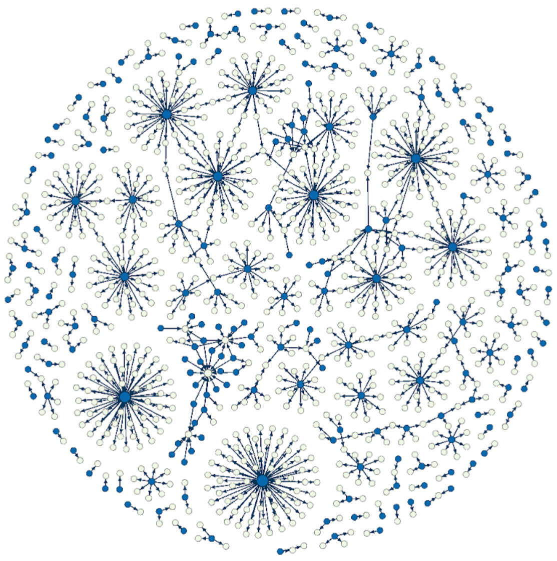
Application-specific CDR information can include the following:

- Account numbers or other internal information; for example, a bank might track which account a caller tried to access in the call.
- IVR information: a sophisticated CDR system can create additional records per call, tracking the individual steps (call flow) that a caller traverses through in the IVR.
- Location: Mobile networks usually track and include base station and cell tower information for a call in their CDR information.

All of this data represents a rich source of information for inferring patterns in user behavior. Finding these patterns, however, requires representing the data in creative ways so that we can visualize and extract features of interest. For example, graphs provide us with a great representation of this data.

Representing CDR data using graphs

A graph is essentially a network with nodes connected by edges. Edges can be directed or undirected. Both nodes and edges can be annotated with information. While they are simple to define, graphs are a powerful way to represent heterogeneous pieces of information which are related to each other, and to understand complex interactions between them.



For example, the figure shows a graph representation of a small sample of CDR data with source and destination ANIs as nodes. The edges are directed, representing a call made from the source to the target. The edges can be annotated with information about the number, duration and timings of the calls. The nodes can be annotated with account numbers and other user information.

Right away, the graph shows us some interesting properties

- Degree: Most source ANIs (in blue) call 1-3 destination ANIs. However, a small number of ANIs call large number of destinations. Similarly, most destinations are called only by a single source, with a few exceptions. What do these exceptions represent?
- Clustering and connectivity: The graph is clearly split into many small and a few large connected components. What do they represent?

These observations about graphs can be represented as numeric values, which we can use as features in further statistical and machine learning analysis.

Feature design for CDR analysis

Features, in machine learning terminology, are metrics which, statistically speaking, are likely to distinguish good and bad instances. For example, our audio analysis uses 147 features extracted from the audio. In the case of CDRs, we extract three categories of features: reputation, velocity and behavior.

Reputation features

When we get a phone call, the most obvious questions to ask are: What do we know about the calling phone number? Has it been already associated with suspicious or malicious activity? We track several pieces of relevant information including: Carrier, Device type (landline, cell, VOIP) and complaints associated with this number, both in public and private databases. Using these and other data, we calculate a risk score for every phone number.

Velocity features

Velocity features summarize the trends associated with an ANI over time. Many velocity features emerge naturally from the graph representation, e.g.: the number of destination ANIs that a source ANI has called, the number of user accounts associated with an ANI, the average frequency of calls from or to an ANI. All of these features are essentially graph properties: node degree, edge creation frequency and so on.

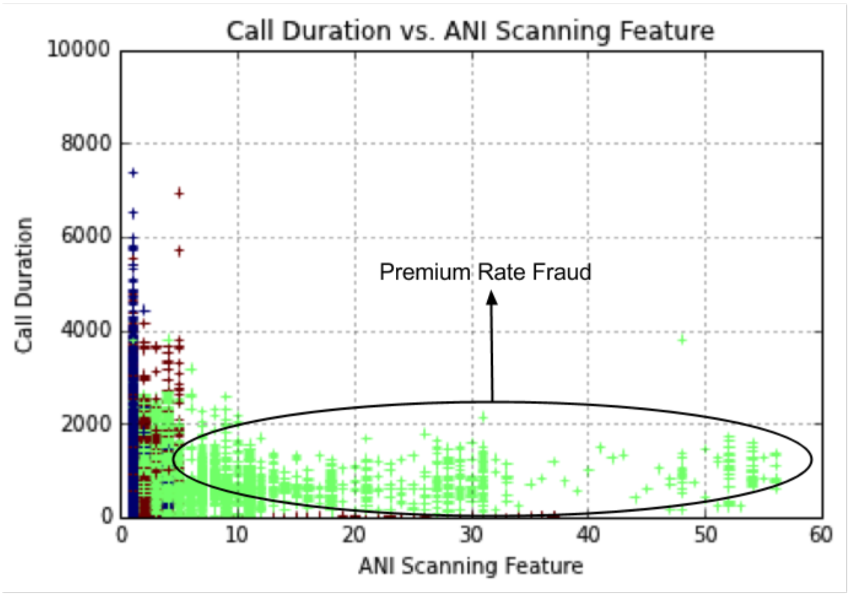
Behavior features

Behavior features capture how a caller acts during a specific call, particularly in the IVR. We first break up the IVR call flow sequence into short chunks, then convert each chunk into a fixed-length, numeric feature vector. This can lead to very high-dimensional feature vectors, so we select the top K features using a feature selection technique such as the chi-square method.

How well do these features perform in machine learning? We use two case studies to illustrate.

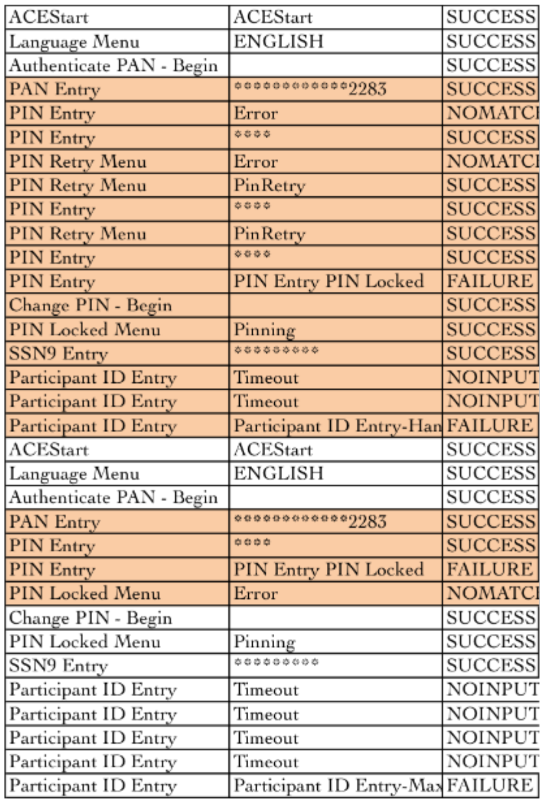
Case study 1: Calling card telecommunication provider

A major calling card company had a problem of *premium rate fraud*; a fraudster uses stolen calling card numbers to make calls to fake premium numbers in other countries, pocketing the fees. We realized that the fraudsters were using automated robots, both to discover valid calling cards (ANI scanning), and to actually call using those cards. By designing features based on graph analysis, along with features capturing duration of calls, interval between successive calls and periodicity of calls from a source, we detected over 80% of the premium rate fraud; in some cases, up to 10 days before actual fraud calls. We also created a custom feature to detect ANI scanning by identifying sequences of attempts using similar ANIs. Our ANI scanning feature identified 50% of the premium rate fraud, as the figure below shows.

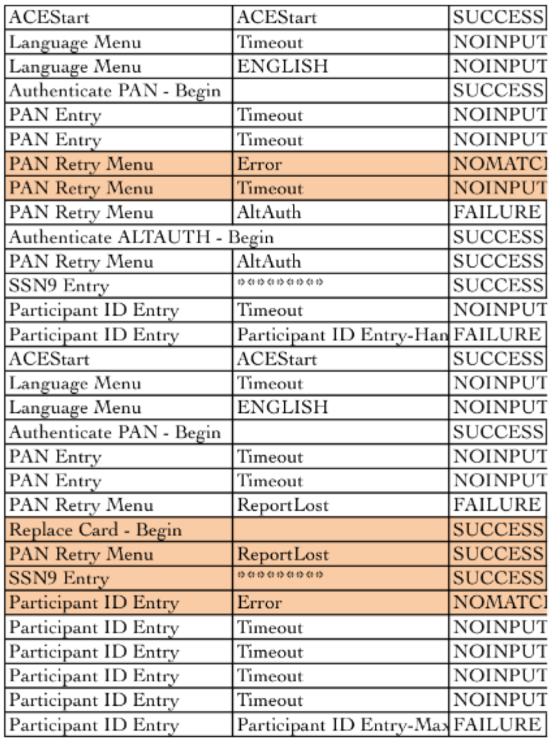


Case study 2: State benefits provider

A major state benefits had fraudsters filing fraudulent claims causing major losses. In particular, fraudsters were using the IVR system to find valid information, which they then used to take over accounts. In order to detect this kind of probing activity, we combined all three categories of features, trained a machine learning model on a labeled subset of data and used it to score new calls.



ACEStart	ACEStart	SUCCESS
Language Menu	ENGLISH	SUCCESS
Authenticate PAN - Begin		SUCCESS
PAN Entry	************2283	SUCCESS
PIN Entry	Error	NOMATCI
PIN Entry	****	SUCCESS
PIN Retry Menu	Error	NOMATCI
PIN Retry Menu	PinRetry	SUCCESS
PIN Entry	****	SUCCESS
PIN Retry Menu	PinRetry	SUCCESS
PIN Entry	****	SUCCESS
PIN Entry	PIN Entry PIN Locked	FAILURE
Change PIN - Begin		SUCCESS
PIN Locked Menu	Pinning	SUCCESS
SSN9 Entry	*********	SUCCESS
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Participant ID Entry-Han	FAILURE
ACEStart	ACEStart	SUCCESS
Language Menu	ENGLISH	SUCCESS
Authenticate PAN - Begin		SUCCESS
PAN Entry	************2283	SUCCESS
PIN Entry	****	SUCCESS
PIN Entry	PIN Entry PIN Locked	FAILURE
PIN Locked Menu	Error	NOMATCI
Change PIN - Begin		SUCCESS
PIN Locked Menu	Pinning	SUCCESS
SSN9 Entry	*********	SUCCESS
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Participant ID Entry-Max	FAILURE



ACEStart	ACEStart	SUCCESS
Language Menu	Timeout	NOINPUT
Language Menu	ENGLISH	NOINPUT
Authenticate PAN - Begin		SUCCESS
PAN Entry	Timeout	NOINPUT
PAN Entry	Timeout	NOINPUT
PAN Retry Menu	Error	NOMATCI
PAN Retry Menu	Timeout	NOINPUT
PAN Retry Menu	AltAuth	FAILURE
Authenticate ALTAUTH - Begin		SUCCESS
PAN Retry Menu	AltAuth	SUCCESS
SSN9 Entry	*********	SUCCESS
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Participant ID Entry-Han	FAILURE
ACEStart	ACEStart	SUCCESS
Language Menu	Timeout	NOINPUT
Language Menu	ENGLISH	NOINPUT
Authenticate PAN - Begin		SUCCESS
PAN Entry	Timeout	NOINPUT
PAN Entry	Timeout	NOINPUT
PAN Retry Menu	ReportLost	FAILURE
Replace Card - Begin		SUCCESS
PAN Retry Menu	ReportLost	SUCCESS
SSN9 Entry	*********	SUCCESS
Participant ID Entry	Error	NOMATCI
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Timeout	NOINPUT
Participant ID Entry	Participant ID Entry-Max	FAILURE

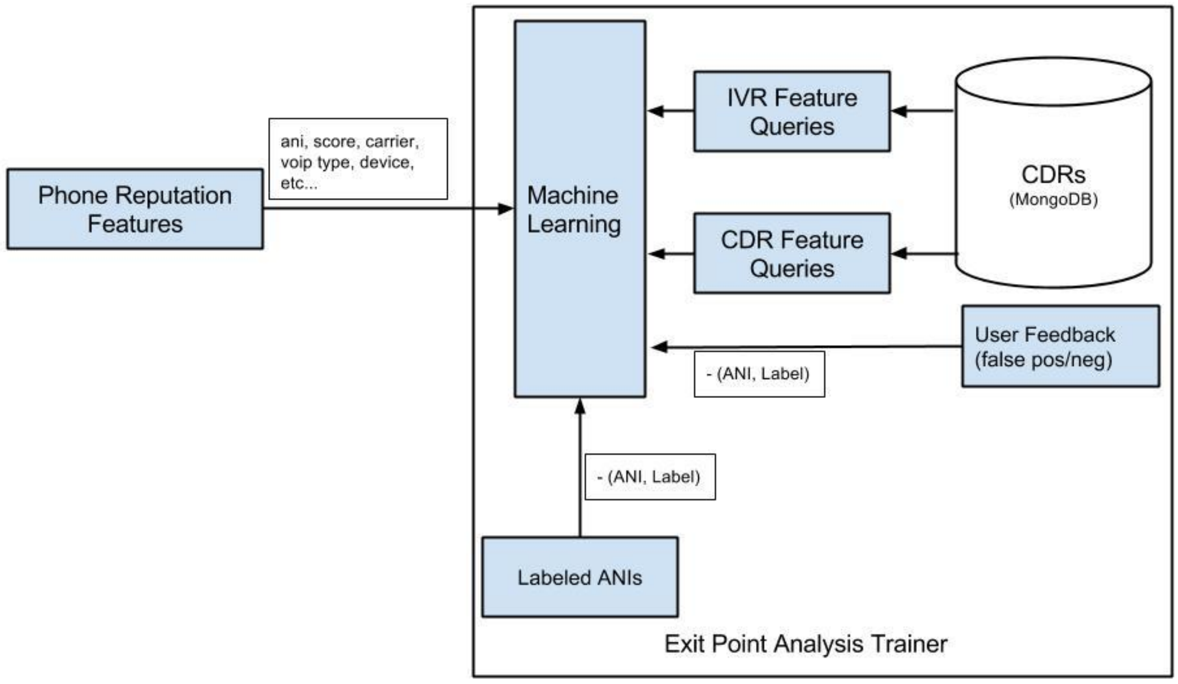
The figure shows an example of IVR reconnaissance that our system identified. A fraudster makes 9 calls from the same ANI (760-538-XXXX) in a space of 10 minutes. The call flow sequence sample shows several attempts at entering a valid PIN number, followed by a card replacement attempt. This is only a small sample of the call flow sequence, but the rest follows a similar pattern.

One of the main challenges we ran into with this use case was the sheer volume of data. Our state benefits provider handles about 20 million calls per month, leading to hundreds of gigabytes of CDR data. How can we perform feature extraction and machine learning on such a high-volume data set?

Analysis at Scale

The first revision of CDR analysis used python data science tools such as pandas and ipython to generate CDR and IVR features. This required that we process only a sample of the larger dataset. We decided to try MongoDB to store CDR data and to query for features used to build risk prediction models. After switching to this implementation, we were able to build prediction models using more than 100 million records.

We chose MongoDB based on several requirements. It has a Python driver 'pymongo' that interoperates with the python data science stack. It is schemaless so we could ingest CDR data with arbitrary formats. MongoDB (like many data stores) provides a bulk insert API that allows us to insert thousands of CDRs in a single API call. Finally, Aggregations API provides a flexible search api that supports query parallelism and horizontal scalability.


Phone Reputation Features
ani, score, carrier, voip type, device, etc...
Machine Learning
IVR Feature Queries
CDR Feature Queries
CDRs (MongoDB)
User Feedback (false pos/neg)
- (ANI, Label)
- (ANI, Label)
Labeled ANIs
Exit Point Analysis Trainer

Data Ingest

A key determinant of write throughput is the “write concern” setting. Write concern describes the guarantee that MongoDB provides when reporting on the success of a write operation [1]. Operationally this means: the greater the guarantee, the slower insert throughput. We use the “journaled write concern” for bulk insert operations to guarantee that CDRs are fully committed to disk after each insert. For updates, we use the lesser “acknowledged write concern” that simply guarantees the DB server received the request. For a single node MongoDB cluster and the aforementioned settings, we saw insert speeds of 8000 records per second and updates of 1000 records per second.

Generating CDR and IVR Features

The Aggregations Framework provides a flexible API to query for IVR and CDR features used to build risk prediction models. The figure below shows a simple query to count the number of unique accounts associated with a particular phone number.

Aggregation queries are constructed as a data processing pipeline. Line 2 selects "ANI" and account ids fields from the records. Lines 3 - 7 generate group objects where the key is ANI and value is a set of unique account numbers. Line 8 generates a cross-product between the "ANI" and account ids. Finally, line 9 generates group objects where the key is ANI and value is COUNT of “ACCOUNT IDS”.

The pymongo api is thread-safe, so we parallelized the aggregation queries using multiprocessing.dummy.pool [3]. The pool distributes queries into groups of ANIs to be queried by the each thread concurrently. Each added thread provided a linear reduction in query latency.

```
db.cdr.aggregate(
      {"$project" : {"ANI": 1, "ACCOUNT_ID":1}},
      {"$group" :
         {"_id": "$"+"ANI",
          "value_set":{"$addToSet":"$"+"ACCOUNT_ID"}
         }
      },
      {"$unwind": "$value_set"},
      {"$group": {"_id": "$_id", "num_unique": {"$sum": 1}}},
);
```

Storing Feature Vectors

The CDR features collected from queries are used to generate risk prediction models using scikit-learn. A useful way to (ab)use MongoDB is to store normalized feature vectors as Binary BSON data. This allows our prediction processes to reuse the feature vectors for subsequent experiments. The below figure demonstrates how to store serialized feature vectors to MongoDB.

```
# … code above performs feature selection and normalization
ret = db.cdr.save({'time': datetime.datetime.now(),
                'X': bson.binary.Binary(
                      cPickle.dumps(self.X, cPickle.HIGHEST_PROTOCOL)),
                'y': bson.binary.Binary(
                      cPickle.dumps(self.y, cPickle.HIGHEST_PROTOCOL))
})
```

The below figure shows how to load the feature vectors from MongoDB and loading them into a scikit-learn classifier.

```
doc = db.cdr.find({}).sort("time", -1).limit(1)[0]
X = cPickle.loads(doc['X'])
y = cPickle.loads(doc['y'])
clf.fit(X, y)
# … code after performs clf.predict()
```

Conclusion

In this whitepaper we show how you can identify a fraudster using acoustical anomalies. In addition, with CDR analysis you can identify when he is performing reconnaissance activity before the actual attack. These two techniques allow us to provide a detailed lifecycle of a phone fraudster.