Account Takeover The Solution:
Account takeover comes in many forms capable of bypassing most existing authentication and fraud prevention controls. Behavioral biometrics looks for anomalies in digital interactions that can indicate both human and nonhuman attacks such as malware, bots, remote access tools and manual account takeover methods.

Techniques used by the scammers to commit fraud:

- Smishing
- Voice Scams
- Remote Access Scams

Legacy-based fraud prevention measures that rely on device, IP or network attributes are no longer a match on their own for scammers who have learned to bypass or spoof them, and new approaches that provide deeper visibility into risk across a digital session are required.

Vulnerable and Targete Customers:

- Elderly Customer
- Generation Z

Industry groups, such as The Knoble, have also been formed to identify best practices for detection and data sharing requirements within financial institutions to combat financial crimes such as elder scams. Technology that leverages machine learning and an understanding of human behavior will be a critical element in protecting vulnerable customer populations.

Mobile Malware
Remote access capabilities are common in financial malware. Specifically, TeaBot leverages RAT capabilities that enable keylogging, interception of OTP codes and harvesting of other data contained on an infected device.
Scammers coerce users into using one of these tools (Zoom, Any Desk and Team Viewer) and sharing their screens so that the scammer can more effectively coach them through a scam. BioCatch refers to this attack method as a Passive Remote Access scam and has discovered over 80% of iOS reported remote access cases contain this positive screen broadcast feature. This means that the passive remote access was being broadcast to multiple screens to better complete the scam process.
As malware behaves in ways much different than the genuine user population, and these indicators are common in most malware families, behaviors indicative of malware can be uncovered in actions such as navigation paths, accelerometer data, and touch and swipe patterns.


BioCatch Account Takeover Protection continuously monitors web and mobile sessions for user application, behavioral, device, and network anomalies and applies advanced risk models to expose a broad range of account takeover threats that legacy fraud prevention controls miss.

BioCatch Account Takeover Protection:
Analyzing Patterns in Human Activity:

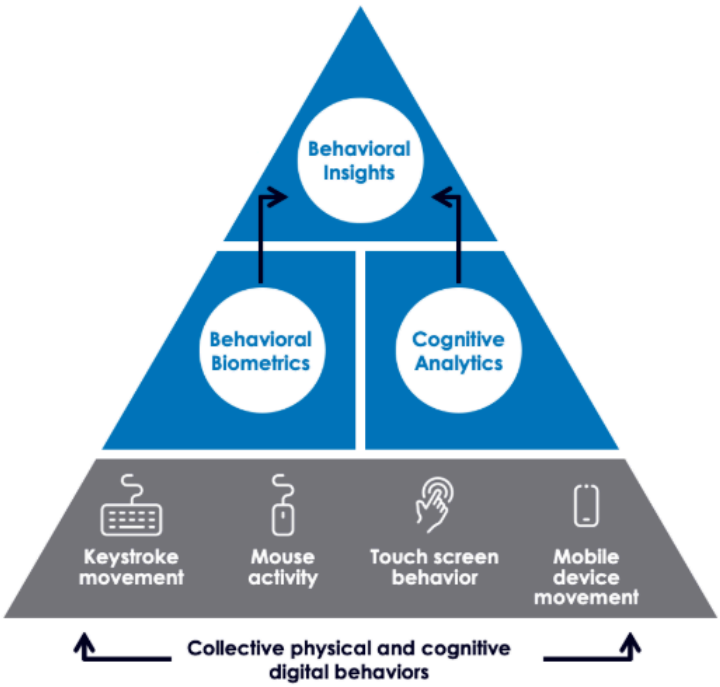
- Mouse Activity (Speed-Movement Patterns - Scroll Preferences)
- Keystroke Movement (Speed-Shortcuts-Advanced Keys)
- Touchscreen Behaviour (Press Size - Area - Pressure)
- Device Movement (Gyro ~~Ghyro~~ - Orientation - Scrolling)

BioCatch risk models leverage innovative research and a decade of behavioral data to distinguish between genuine users and cybercriminals. Using a multi-layered approach to analysis, BioCatch empowers organizations to detect more fraud without worrying about disrupting the experience for genuine customers.

The BioCatch platform determines a session's risk level by analyzing a user's digital behavior on three levels:

- Behavioral Biometrics
- Cognitive Analysis
- Behavioral Insights

Behavioral Biometrics: ‘
The BioCatch platform compares a user’s physical behavior, including navigation preferences, hand-eye coordination, and press size, against the user’s historical profile to detect anomalies, including human versus automated or bot activity.
Cognitive Analysis:
The BioCatch platform compares a user’s cognitive behavior, including the use of shortcuts, long-term memory, and navigation patterns, against population-level profiles to identify genuine and criminal behavioral patterns.
Behavioral Insights:
The BioCatch platformdetermines user intent and emotional state by surfacing behaviors that align with complex fraud threats, such as social engineering voice scams, including hesitation, dictation, duress, and more.



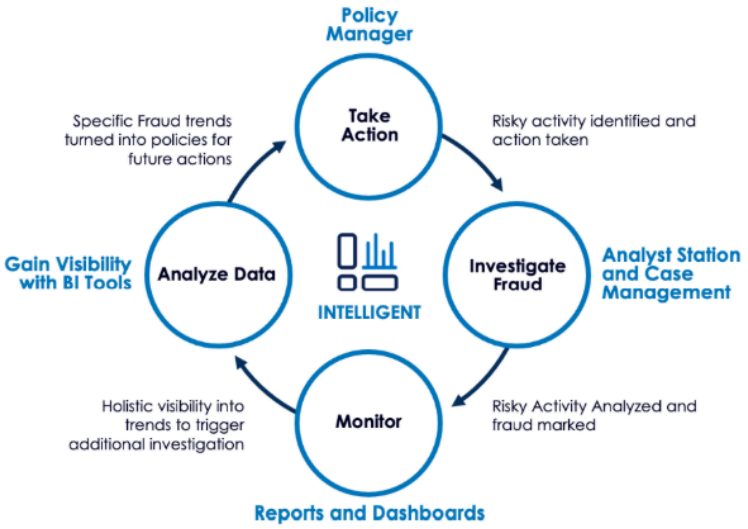

Through continuous session visibility and advanced profiling capabilities, the BioCatch platform detects even the most subtle signs of criminal activity, providing a greater breadth of coverage against the most sophisticated account takeover attacks including:

- Bots & Aggregators
- Malware
- Remote Access Tools
- Mobile Emulators
- Social Engineering Voice Scams
- Stolen Account Credentials

The BioCatch solution detects key behavioral indicators that align with popular attack methods,such as automated patterns which can indicate account takeover via bot or malware, and surfaces anomalies that can suggest a human imposter, such as dominant hand change and expert user patterns.

- Reduce Fraud. Increase Trust.
  Effectively manage risk across digital channels to build customer trust and confidence
- Instantly Gain Actionable Insights
  Leverage unique behavioral insights to determine the appropriate course of action
- Improve Customer Experience
  Significantly reduce false positives and abandonment to banish friction and boost user experience

BioCatch Account Takeover Protection delivers profound visibility into fraud risk through a powerful set of **platform tools**; these tools enable fraud teams to drive real-time action and investigate fraud incidents with ease. Organizations that prefer to **perform their own analysis** can consume **BioCatch risk scores**, **threat** and **genuine indicators**, and raw data via **API**.



# Implementation & Tech-stack to use:

### Steps to Implement Behavioral Biometrics

1. Data Collection:
    - Navigation Preferences: Track mouse movements, scrolling patterns, page transitions, etc.
    - Hand-Eye Coordination: Capture mouse pointer accuracy, reaction times, and other related metrics.
    - Press Size: Record keystroke dynamics, including pressure, duration, and intervals.
    -
2. Data Processing:
    - Normalize and preprocess the collected data to ensure consistency and remove noise.
    - Feature extraction to identify key metrics that can be used for analysis.

3. Profile Creation:
    - Develop historical profiles for each user based on the collected data.
    - Use statistical models or machine learning algorithms to represent typical user behavior.

4. Anomaly Detection:
    - Implement algorithms to compare current behavior against the historical profile.
    - Use techniques like distance metrics (e.g., Euclidean distance), clustering, or more advanced ML models.

5. Human vs. Bot Detection:
    - Train models specifically to distinguish between human and automated activities.
    - Employ techniques like supervised learning, anomaly detection models, or heuristic rules.

### Tech Stack

1. Data Collection
    - Frontend: JavaScript, HTML5, CSS3
    - Libraries: jQuery (for DOM manipulation), D3.js (for data visualization)
    - Tools: Mouseflow, FullStory (for user behavior tracking)

2. Backend
    - Server: Node.js, Python (Flask or Django)
    - Libraries: Express (for Node.js), Pandas and NumPy (for Python)
    - Databases: MongoDB, PostgreSQL

3. Machine Learning & Data Processing

- Python: scikit-learn, TensorFlow, PyTorch
- Data Processing: Apache Kafka (for real-time data streaming), Apache Spark (for large-scale data processing)

4. Anomaly Detection Algorithms
    - Supervised Learning: Random Forest, Support Vector Machines, Neural Networks
    - Unsupervised Learning: K-means Clustering, DBSCAN, Isolation Forest
    - Distance Metrics: Euclidean, Manhattan

5. Deployment
    - Cloud Providers: AWS, Google Cloud, Azure
    - Services: AWS Lambda (for serverless processing), Google Cloud Functions, Azure Functions
    - Containers: Docker, Kubernetes

## Steps to Implement Cognitive Analysis

1. Data Collection:
 - Use of Shortcuts: Track keyboard shortcuts usage, mouse shortcuts, and command patterns.
 - Long-term Memory: Analyze repeated actions over time, recall patterns, and user-specific sequences.
 - Navigation Patterns: Monitor page navigation paths, click sequences, and time spent on tasks.

2. Data Processing:
 - Preprocess and clean the collected data to ensure accuracy and remove inconsistencies.
 - Extract relevant features such as frequency of shortcuts, navigation path regularity, and task completion times.

3. Profile Creation:
 - Develop individual profiles based on cognitive behavior data.
 - Aggregate data to create population-level profiles for comparison.

4. Pattern Analysis:
 - Compare individual user profiles with population-level profiles to identify anomalies.
 - Use statistical and machine learning models to detect patterns indicative of genuine or criminal behavior.

5. Behavioral Identification:
 - Implement algorithms to classify behaviors as genuine or potentially malicious.
 - Continuously update models with new data to improve accuracy and adaptability.

## Tech Stack

1. Data Collection
   - Frontend: JavaScript, HTML5, CSS3
   - Libraries: React.js (for interactive UIs), jQuery (for DOM manipulation)
   - Tools: Hotjar (for user behavior tracking), Google Analytics

2. Backend
   - Server: Node.js, Python (Flask or Django)
   - Libraries: Express (for Node.js), Pandas and NumPy (for Python)
   - Databases: MongoDB, PostgreSQL

3. Machine Learning & Data Processing
   - Python: scikit-learn, TensorFlow, PyTorch
   - Data Processing: Apache Kafka (for real-time data streaming), Apache Spark (for large-scale data processing)

4. Pattern Analysis Algorithms
   - Supervised Learning: Random Forest, Support Vector Machines, Neural Networks
   - Unsupervised Learning: K-means Clustering, DBSCAN, Isolation Forest
   - Time Series Analysis: ARIMA, LSTM Networks

5. Deployment
   - Cloud Providers: AWS, Google Cloud, Azure
   - Services: AWS Lambda (for serverless processing), Google Cloud Functions, Azure Functions
   - Containers: Docker, Kubernetes

### Implementation Example

1. Data Collection:
   - Use JavaScript to capture user interactions and send data to the backend.

**Javascript**

```
document.addEventListener('keydown', function(event) {
    fetch('/collect_data', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ key: event.key, timestamp: Date.now() })
    });
});
```

2. Backend (Flask):
   - Receive and store the data, preprocess it, and perform feature extraction.

**Python**

```
from flask import Flask, request
import pandas as pd

app = Flask(__name__)
data = []

@app.route('/collect_data', methods=['POST'])
def collect_data():
    global data
    event = request.get_json()
    data.append(event)
    return '', 200

@app.route('/process_data')
def process_data():
    df = pd.DataFrame(data)
    # Perform preprocessing and feature extraction
    # Example: df['interval'] = df['timestamp'].diff()
    # Save or use the processed data for ML model training
    return 'Data Processed', 200

if __name__ == '__main__':
    app.run(debug=True)
```

3. Machine Learning (TensorFlow):
   - Train a model to detect anomalies or classify behaviors.

**Python**

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Assume `processed_data` is a DataFrame with features
X = processed_data[['feature1', 'feature2', 'feature3']]
y = processed_data['label']

model = Sequential([
    Dense(64, activation='relu', input_shape=(X.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=10, batch_size=32)
```

```
# Save the model
model.save('cognitive_model.h5')
```

## Steps to Implement Behavioral Insights

1. Data Collection:
   - Voice Data: Capture audio data from phone calls or voice interactions.
   - Interaction Data: Collect metadata such as call duration, pauses, and response times.

2. Feature Extraction:
   - Acoustic Features: Extract features like pitch, tone, speech rate, pauses, and volume.
   - Linguistic Features: Analyze text transcripts for hesitation words, dictation patterns, and signs of duress.

3. Data Processing:
   - Normalize and preprocess the audio and text data.
   - Apply feature extraction techniques to convert raw data into meaningful features.

4. Model Training:
   - Train machine learning models to identify patterns associated with fraud, social engineering, and emotional states.
   - Use supervised learning with labeled datasets to train models to detect specific behaviors.

5. Anomaly Detection:
   - Implement algorithms to detect anomalies in voice and interaction patterns.
   - Use unsupervised learning or clustering techniques to identify outliers.

6. Behavioral Analysis:
   - Compare real-time data against trained models to determine user intent and emotional state.
   - Surface behaviors indicative of complex fraud threats such as hesitation, dictation under duress, etc.

## Tech Stack

1. Data Collection
   - Voice Capture: Twilio, Nexmo (for phone calls), WebRTC (for web-based voice capture)
   - Audio Processing: Python, JavaScript
   - Libraries: Pydub, Wave

2. Backend
   - Server: Node.js, Python (Flask or Django)
   - Libraries: Express (for Node.js), Flask-RESTful (for Python)

- Databases: MongoDB, PostgreSQL

3. Machine Learning & NLP
   - Python: scikit-learn, TensorFlow, PyTorch
   - Libraries: librosa (for audio analysis), NLTK, SpaCy (for text processing)
   - Speech Recognition: Google Speech API, IBM Watson, Azure Speech to Text

4. Feature Extraction
   - Audio Features: librosa, Praat
   - Text Features: NLTK, SpaCy, TextBlob

5. Anomaly Detection
   - Algorithms: Isolation Forest, DBSCAN, One-Class SVM
   - Libraries: scikit-learn, PyOD

6. Deployment
   - Cloud Providers: AWS, Google Cloud, Azure
   - Services: AWS Lambda, Google Cloud Functions, Azure Functions
   - Containers: Docker, Kubernetes

## Implementation Example

1. Data Collection:
   - Use Twilio to capture voice data from phone calls.

**Python**

```
from flask import Flask, request, jsonify
import twilio.twiml

app = Flask(__name__)

@app.route('/voice', methods=['POST'])
def voice():
    resp = twilio.twiml.Response()
    resp.say("Please speak after the beep.")
    resp.record(maxLength="30", action="/handle-recording")
    return str(resp)

@app.route('/handle-recording', methods=['POST'])
def handle_recording():
    recording_url = request.form['RecordingUrl']
    # Process the recording
    return "Recording received", 200

if __name__ == "__main__":
```

```
    app.run(debug=True)
```

2. Feature Extraction:
   - Extract acoustic and linguistic features from the recorded audio.

**Python**

```
import librosa
import numpy as np

def extract_features(file_path):
    y, sr = librosa.load(file_path, sr=None)
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    mfccs_mean = np.mean(mfccs.T, axis=0)
    return mfccs_mean
```

3. Model Training:
   - Train a model to detect emotional states and potential fraud patterns.

**Python**

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Assume `X_train` is the feature matrix and `y_train` are the labels
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Save the model
model.save('behavioral_model.h5')
```

4. Real-Time Analysis:
   - Analyze incoming data in real-time to determine user intent and emotional state.

**Python**

```
from tensorflow.keras.models import load_model

model = load_model('behavioral_model.h5')
```

```
def analyze_behavior(file_path):
    features = extract_features(file_path)
    features = features.reshape(1, -1)
    prediction = model.predict(features)
    return prediction

@app.route('/analyze', methods=['POST'])
def analyze():
    file_path = request.form['file_path']
    prediction = analyze_behavior(file_path)
    return jsonify({'prediction': float(prediction[0][0])})
```

## Resources:

- Behavioral Biometrics
    - "Behavioral Biometrics: A Remote Access Approach" by Kenneth Revett
    - Continuous Authentication Using Behavioral Biometrics
    - User Authentication Based on Mouse Dynamics Using Deep Neural Networks: A Comprehensive Study
    - Authentication of Smartphone Users Using Behavioral Biometrics
    - Open Source Projects & Tools:
        1. Mouseflow
        2. Keystroke Dynamics
        3. Pandas
        4. Scikit-learn

- Cognitive Analysis
    - Evaluating Behavioral Biometrics for Continuous Authentication: Challenges and Metrics
    - "Applied Predictive Modeling" by Max Kuhn and Kjell Johnson
    - A Review of EEG-Based User Authentication: Trends and Future Research Directions
    - User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder - Insider Threat Detection

- Behavioral Insights
  - Pattern Recognition and Machine Learning
  - Stress and Deception in Speech: Evaluating Layered Voice Analysis
  - Analysis of Speech Emotion Recognition and Detection using Deep Learning
  - Deception in Spoken Dialogue: Classification and Individual Differences
  - The Truth and Nothing but the Truth: Multimodal Analysis for Deception Detection
  - Speech and Language Processing by Daniel Jurafsky and James H. Martin
  - Open Source Projects & Tools:
    1. Pydub
    2. librosa
    3. NLTK
    4. SpaCy