08/02/2024 DeepSafe

Next Steps

- [ ] Contact professionals to get first opinion
- [ ] Prototype creation process
    - [ ] List features and core functionalities (user flow)
    - [ ] Diagram or workflow of technical implementation
    - [ ] Create slide deck or doc with features to prioritize
        - [ ] Lists: do it for 07/31/2024
        - [ ] User flow diagram: do it for 31/07/2024
    - [ ] Create diagram with intended functionality
    - [ ] Use ChatGPT to generate code background for initial testing
    - [ ] Begin google sheets the bank of "red flag" conversations in a google sheet
        - [ ] Combination of manual creation (from ourselves, from the textbook): Scenario creation
        - [ ] And ChatGPT auto-generated
    - [ ] Figuring out the accuracy of how good ChatGPT can actually detect transcript conversations

Deck Structure:
Link Miro:
https://miro.com/welcomeonboard/QzJQaTVyN0ZOMVZEZU5oc3pDa3A5OU83eFdYc201a3Y2UTlaaTdWdFM0RjJiYTV4Qmt1RGxnZFAyaGdmdEY4V3wzNDU4NzY0NTc2MjMyMDkxNzAzfDI=?share_link_id=336742322243

Core Features

-

Nice to Have Features

-

User Flow Diagram: Priority
Create a diagram on miro
Share it
Complete it for next week
Review it during 07/31/2024's Meeting

Technical Flow Diagram

MFA (Multi factor authentication)

Multi Factor authentication via messaging for DeepFake detection. Analyze what is said in the meeting transcript translated to text for chatgpt analysis, send the specific instruction for chatGPT, then send a notification to the smartphone of the people in the meeting. ChatGPT does

- Transcript analysis
- Social Engineering manipulation

Partnership: collaborating with a company to develop standard scenarios and specific scenarios for specific scenarios.

Steps

1. Detect suspicious activity about money transfer
2. Send a text message to the CEO about confirming if they are in the meeting
3. Have them say “yes” or “no” to verify their presence in that zoom meeting
4. If yes
   a. Then ???
5. If no
   a. Send a text message to the participant to alert them about the suspicious fake other person

Solution Overview

1. Meeting Transcript Collection: Capture audio and video from the meeting and convert it into text using speech-to-text technology.
2. Deepfake Detection: Analyze the transcript for authenticity using deepfake detection models.
3. ChatGPT Analysis: Send the transcript to ChatGPT for further analysis, including checking for inconsistencies or unusual patterns.
4. Notification System: Send a notification to the participants' smartphones with the results of the analysis.

Steps and Components

1. Meeting Transcript Collection

- Capture Audio and Video: Use a service like Zoom or Microsoft Teams to record the meeting.
- Speech-to-Text Conversion: Use APIs like Google Cloud Speech-to-Text or AWS Transcribe to convert the audio to text.

2. Deepfake Detection

- **Latency and QoS Data Analysis**: Extract and analyze audio-video latency using QoS data to detect synchronization issues indicative of deepfakes.
- **Deepfake Detection Model**: Utilize pre-trained models or custom models trained on datasets like FaceForensics++ to analyze the video frames for deepfake signs.

3. ChatGPT Analysis

- **Extract Key Points from Transcript**: Use natural language processing (NLP) techniques to extract important points and context from the transcript.
- **Send Specific Instructions to ChatGPT**: Format and send specific instructions to ChatGPT for further analysis. This could include checking for logical consistency, detecting unusual patterns, or identifying suspicious statements.

4. Notification System

- **Generate Analysis Report**: Compile the results from the deepfake detection and ChatGPT analysis into a report.
- **Send Notifications**: Use a messaging service (like Twilio or Firebase Cloud Messaging) to send notifications to participants' smartphones.

Implementation Example

Step 1: Meeting Transcript Collection

python
Copy code
```python
import os
from google.cloud import speech_v1p1beta1 as speech
from google.cloud import storage

def transcribe_audio(storage_uri):
    client = speech.SpeechClient()
    audio = speech.RecognitionAudio(uri=storage_uri)
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        language_code="en-US",
    )

    response = client.long_running_recognize(config=config,
audio=audio)
    return response.result()

# Example: transcribe_audio("gs://your-bucket/audio-file.wav")
```

Step 2: Deepfake Detection

python
Copy code
```
def analyze_qos_data(qos_data):
    # Example analysis based on previously trained models
    # Placeholder function to represent deepfake detection
    deepfake_detected = False
    if qos_data['audio_video_latency_diff'] > threshold:
        deepfake_detected = True
    return deepfake_detected
```

Step 3: ChatGPT Analysis
python
Copy code
```
import openai

openai.api_key = 'your-api-key'

def analyze_transcript_with_chatgpt(transcript):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": "You are an assistant that 
analyzes meeting transcripts."},
            {"role": "user", "content": transcript}
        ]
    )
    return response.choices[0].message['content']

# Example: analyze_transcript_with_chatgpt(transcript_text)
```

Step 4: Notification System
python
Copy code
```
from twilio.rest import Client

def send_notification(phone_number, message):
    client = Client('your_account_sid', 'your_auth_token')
    message = client.messages.create(
        body=message,
        from_='+1234567890',
        to=phone_number
    )
    return message.sid

# Example: send_notification("+1234567890", "Deepfake detected in the 
meeting transcript.")
```

Workflow Integration

1. **Capture and Transcribe**: Capture the meeting and transcribe it using speech-to-text.
2. **Analyze for Deepfakes**: Use QoS data and deepfake detection models to analyze the audio-video synchronization.
3. **ChatGPT Analysis**: Analyze the transcript using ChatGPT to detect inconsistencies and unusual patterns.
4. **Notification**: Compile the results and send notifications to participants' smartphones.

Example Workflow Code

python
Copy code
```python
# Transcribe audio
transcript = transcribe_audio("gs://your-bucket/audio-file.wav")

# Analyze QoS data for deepfakes
qos_data = {'audio_latency': 100, 'video_latency': 105,
'audio_video_latency_diff': 5, 'jitter': 10, 'packet_loss': 0.01}
deepfake_detected = analyze_qos_data(qos_data)

# Analyze transcript with ChatGPT
chatgpt_analysis = analyze_transcript_with_chatgpt(transcript)

# Compile report
report = f"Deepfake Detected: {deepfake_detected}\nChatGPT Analysis:
{chatgpt_analysis}"

# Send notification
phone_numbers = ["+1234567890", "+0987654321"]
for number in phone_numbers:
    send_notification(number, report)
```

This solution provides a multifactor authentication approach by combining DeepFake detection, advanced AI analysis, and real-time notifications to ensure the integrity of virtual meetings.

DeepFake Detection

deepfake detection using audio video latency through zoom call QSS data

Deepfake detection using audio-video latency through Zoom call Quality of Service (QoS) data involves analyzing the synchronization between audio and video streams to identify discrepancies that might indicate the presence of deepfake technology. Here are the key aspects and steps involved in this approach:

Key Concepts

1. Deepfake Technology: Uses AI to create realistic but fake audio and video content. Detecting deepfakes is critical for maintaining trust in digital communications.
2. Audio-Video Latency: Refers to the delay between the audio and video streams during transmission. Natural conversations have minimal latency, whereas deepfakes often exhibit noticeable synchronization issues.
3. Zoom QoS Data: Zoom provides Quality of Service (QoS) data, which includes metrics on audio and video performance, network conditions, and latency.

Detection Process

1. Data Collection: Gather QoS data from Zoom calls, focusing on audio and video latency metrics. This data can be accessed through Zoom's dashboard or APIs.
2. Synchronization Analysis: Analyze the synchronization between audio and video streams. Deepfakes often have a noticeable delay or jitter that deviates from the norm.
3. Threshold Setting: Establish baseline thresholds for acceptable audio-video latency. Any deviations beyond these thresholds could be indicative of deepfake content.
4. Machine Learning Models: Train machine learning models to detect patterns associated with deepfake videos. Features can include audio-video latency, jitter, packet loss, and other QoS metrics.
5. Real-Time Monitoring: Implement real-time monitoring of Zoom calls using the trained models to flag potential deepfake content.
6. Validation and Feedback: Continuously validate and refine the detection models with new data and feedback to improve accuracy.

Challenges

1. Data Quality: Ensuring high-quality and comprehensive QoS data is critical for accurate analysis.
2. Adaptability: Deepfake technology evolves rapidly, requiring constant updates to detection models.
3. False Positives/Negatives: Balancing sensitivity and specificity to minimize false alarms while detecting actual deepfakes.

Implementation Example

1. Data Extraction:
    - Use Zoom API to extract QoS data during calls.
    - Focus on metrics such as `audio_latency`, `video_latency`, `jitter`, and `packet_loss`.
2. Feature Engineering:
    - Calculate the time difference between corresponding audio and video packets.
    - Aggregate metrics over time windows (e.g., per second) to identify persistent latency issues.
3. Model Training:
    - Label datasets with known deepfake and real calls.
    - Train a classifier (e.g., Random Forest, SVM, or Deep Learning) using extracted features.
4. Detection and Alerting:
    - Deploy the model in a real-time monitoring system.
    - Set up alerts for detected anomalies in audio-video synchronization.

Training Data

Yes, there is a significant amount of training data for deepfakes available on the internet, which can be used for both creating and detecting deepfake content. Here are some key sources and considerations:

Public Datasets

1. FaceForensics++: This dataset contains a large collection of real and manipulated videos, including deepfakes created using various methods. It is widely used for training and evaluating deepfake detection algorithms.
2. DeepFake Detection Challenge Dataset (DFDC): Provided by Facebook, this dataset includes thousands of deepfake videos along with real videos. It is designed to encourage the development of deepfake detection techniques.
3. Google Deepfake Detection Dataset: Google released a dataset containing both real and deepfake videos to support research in detecting manipulated media.
4. Celeb-DF: A high-quality deepfake video dataset with celebrity faces, created to address the limitations of existing datasets and improve the training of deepfake detection models.

Example Sources for Datasets

- FaceForensics++
- DeepFake Detection Challenge Dataset
- Google Deepfake Detection Dataset
- Celeb-DF

Flagging & Decision Mechanism

Flagging Mechanisms

1. Scenario Structure Detection
    - **Template Matching:** Create templates for common social engineering scenarios (e.g., urgent fund transfer, emergency vendor payment). Use natural language processing (NLP) to match conversation structures against these templates.
    - **Behavioral Analysis:** Identify unusual conversation patterns, such as immediate requests for large fund transfers without prior context.
2. Conversation Detection
    - **Keyword Analysis:** Monitor conversations for keywords and phrases commonly used in social engineering attacks (e.g., "urgent," "manual transfer," "system down").
    - **Contextual Analysis:** Use AI to understand the context of conversations, flagging those that involve financial transactions or confidential information unexpectedly.
3. Scenario Example Detection
    - **Database of Known Scenarios:** Maintain a database of documented social engineering scenarios and use machine learning to detect conversations that match these examples.
    - **Similarity Scoring:** Implement algorithms to score conversations based on their similarity to known attack scenarios, flagging high-scoring conversations for review.
4. Mismatch Detection Between Internal and Meeting Data
    - **Participant Verification:** Cross-check meeting participants against the company's internal directory. Flag discrepancies such as unknown participants or altered email addresses.
    - **Impersonation Alerts:** Detect and flag meetings where a participant's name or email closely resembles that of a high-ranking company official but with slight variations (e.g., `john.doe@companv.com` vs. `john.doe@company.com`).
5. Metadata Detection
    - **Unusual Meeting Times:** Flag meetings scheduled outside of normal business hours, especially those involving financial transactions or high-level executives.
    - **Meeting Frequency:** Monitor the frequency and timing of meetings involving financial discussions. Flag those that occur suddenly or with unusual regularity.
6. Additional Mechanisms
    - **Behavioral Anomalies:** Use machine learning to detect anomalies in participants' behavior during meetings, such as unusual speech patterns or tone changes.

    - **Multi-Factor Verification:** Implement multi-factor authentication for meeting participants, especially for high-stakes meetings involving financial or sensitive information.
    - **Pre-meeting Alerts:** Automatically notify executives and relevant personnel before meetings flagged as high risk, requiring additional verification steps before proceeding.
    - **Post-meeting Review:** Automatically flag and review recordings of meetings that involved requests for financial transactions or confidential information.
    - **Employee Training and Awareness:** Continuously train employees to recognize social engineering tactics and encourage them to report suspicious activities promptly.

By combining these mechanisms, organizations can enhance their ability to detect and prevent social engineering attacks during Zoom calls. Integrating advanced technologies like NLP and machine learning, along with robust verification and review processes, will create a comprehensive defense strategy.

Decision Metrics and Reasoning

To determine if this conversation should be flagged, we will apply the proposed flagging mechanisms and metrics. Each mechanism will be assessed for indicators of a social engineering attack.

1. Scenario Structure Detection

- **Template Matching:** The conversation closely matches the structure of known social engineering scenarios where urgency and manual intervention are emphasized. The phrases "immediate assistance," "urgent opportunity," "transfer $60,000 right away," and "our usual payment system is down" are red flags.
- **Behavioral Analysis:** Immediate request for a large fund transfer without prior discussion is suspicious.

Score: 9/10

2. Conversation Detection

- **Keyword Analysis:** The conversation contains several keywords and phrases typical of social engineering attacks: "urgent," "immediate assistance," "transfer $60,000," and "manual."
- **Contextual Analysis:** The context involves a financial transaction, which is unusual without prior notice or additional verification.

Score: 8/10

3. Scenario Example Detection

- **Database of Known Scenarios:** The conversation matches known scenarios where attackers pose as high-ranking officials to request urgent fund transfers.
- **Similarity Scoring:** High similarity to documented social engineering cases.

Score: 9/10

4. Mismatch Detection Between Internal and Meeting Data

- **Participant Verification:** Verify if the email and name of the "Chief Marketing Officer" match the company's internal data. Check for slight variations in email addresses or names.
- **Impersonation Alerts:** If there is any slight mismatch in details (e.g., email domain variations), this should be flagged immediately.

Score: 7/10 (Assuming we need to verify email and name)

5. Metadata Detection

- **Unusual Meeting Times:** Check the timing of the meeting. If it's outside normal business hours, it increases suspicion.
- **Meeting Frequency:** Determine if this is an unusual or unscheduled meeting.

Score: 6/10 (Needs specific metadata)

6. Additional Mechanisms

- **Behavioral Anomalies:** Analyze the tone and urgency of the request. The urgency and insistence on manual transfer are behavioral red flags.
- **Multi-Factor Verification:** Was there an attempt to verify the identity of the CMO through another channel?

Score: 8/10

Overall Assessment

Total Score: 47/60

Given the high score (above 75%), this conversation should be flagged for the following reasons:

1. **High Match to Known Scenarios:** The conversation structure and content match common social engineering tactics.
2. **Use of Urgent Language:** The repeated emphasis on urgency and manual processing is characteristic of social engineering attacks.
3. **Unverified Identity:** Without multi-factor verification, the request's legitimacy is highly questionable.

4. **Potential Mismatch in Internal Data:** Any slight mismatch in email or name details would further justify flagging.

Conclusion

Based on the high metrics across various detection mechanisms, this conversation transcript exhibits multiple red flags indicative of a social engineering attack and should be flagged for further investigation and immediate verification.

Scoring and Decision-Making Process

To generalize the scoring and decision-making process for flagging potential social engineering attacks in Zoom calls, we will develop a standardized framework. This framework involves converting text conversations into numerical metrics and calculating an overall security index score.

Step 1: Define Metrics and Weightings

Identify key metrics for evaluating the conversation:

1. **Scenario Structure Detection (20%)**
2. **Keyword and Phrase Analysis (20%)**
3. **Scenario Example Detection (20%)**
4. **Mismatch Detection Between Internal and Meeting Data (15%)**
5. **Metadata Detection (10%)**
6. **Behavioral Anomalies and Verification (15%)**

Step 2: Convert Text to Numerical Metrics

Use Natural Language Processing (NLP) and machine learning techniques to analyze the text and generate scores for each metric. Here's how each metric can be evaluated:

1. **Scenario Structure Detection:**
   - **Template Matching:** Compare the conversation structure against predefined templates of known attack scenarios.
   - **Scoring:** Assign a score (0-10) based on the similarity percentage to these templates.
2. **Keyword and Phrase Analysis:**
   - **Keyword Matching:** Identify the presence of high-risk keywords (e.g., "urgent," "manual transfer," "system down").
   - **Scoring:** Assign a score (0-10) based on the density and context of these keywords.

3. Scenario Example Detection:
    - **Database Comparison:** Match the conversation to a database of documented social engineering scenarios.
    - **Scoring:** Assign a score (0-10) based on the similarity to known examples.
4. Mismatch Detection Between Internal and Meeting Data:
    - **Participant Verification:** Cross-check participant names and emails against internal records.
    - **Scoring:** Assign a score (0-10) based on the presence of mismatches or anomalies.
5. Metadata Detection:
    - **Timing and Frequency:** Analyze the meeting's timing and frequency for irregularities.
    - **Scoring:** Assign a score (0-10) based on the likelihood of unusual metadata.
6. Behavioral Anomalies and Verification:
    - **Tone and Urgency Analysis:** Detect abnormal urgency and behavioral patterns.
    - **Multi-Factor Verification:** Check for verification steps taken during the conversation.
    - **Scoring:** Assign a score (0-10) based on behavioral red flags and verification presence.

Step 3: Calculate Overall Security Index Score

Combine the individual scores using the predefined weightings to calculate an overall security index score. The formula is:

Security Index Score=∑(Metric Score×Weight)\text{Security Index Score} = \sum (\text{Metric Score} \times \text{Weight})Security Index Score=∑(Metric Score×Weight)

Step 4: Decision Thresholds

Establish thresholds for flagging based on the security index score:

- **Low Risk (0-30):** No flagging needed.
- **Medium Risk (31-60):** Monitor and verify if necessary.
- **High Risk (61-100):** Immediate flagging and investigation.

Example Calculation

Let's apply this process to the provided conversation:

Transcript Analysis:

vbnet
Copy code
```
Attacker (Impersonating Chief Marketing Officer): Hello [Marketing
Coordinator's Name], I need your immediate assistance with a
time-sensitive marketing campaign.
```

```
Attacker: We have an urgent opportunity to promote our new product,
and I need you to transfer $60,000 right away to secure the deal. Our
usual payment system is down, so we need to do this manually.

Attacker: This campaign is crucial for our product launch. Here are
the bank details: [Fake Bank Account Information]. Please handle this
urgently and confirm once the transfer is complete. Thank you.
```

1. Scenario Structure Detection (9/10)
2. Keyword and Phrase Analysis (8/10)
3. Scenario Example Detection (9/10)
4. Mismatch Detection Between Internal and Meeting Data (7/10)
5. Metadata Detection (6/10)
6. Behavioral Anomalies and Verification (8/10)

Weights:

- Scenario Structure Detection: 20%
- Keyword and Phrase Analysis: 20%
- Scenario Example Detection: 20%
- Mismatch Detection: 15%
- Metadata Detection: 10%
- Behavioral Anomalies and Verification: 15%

Calculation: Security Index Score=(9×0.20)+(8×0.20)+(9×0.20)+(7×0.15)+(6×0.10)+(8×0.15)\text{Security Index Score} = (9 \times 0.20) + (8 \times 0.20) + (9 \times 0.20) + (7 \times 0.15) + (6 \times 0.10) + (8 \times 0.15)Security Index Score=(9×0.20)+(8×0.20)+(9×0.20)+(7×0.15)+(6×0.10)+(8×0.15) Security Index Score=1.8+1.6+1.8+1.05+0.6+1.2\text{Security Index Score} = 1.8 + 1.6 + 1.8 + 1.05 + 0.6 + 1.2Security Index Score=1.8+1.6+1.8+1.05+0.6+1.2 Security Index Score=8.05\text{Security Index Score} = 8.05Security Index Score=8.05

Final Decision

Given the Security Index Score of 8.05 (out of 10), this conversation falls into the **High Risk** category and should be flagged for immediate investigation. This process ensures a systematic and consistent approach to identifying and flagging potential social engineering attacks in Zoom calls.