**PROJECT PROPOSAL**

**Group No -01**

*Prepared By*

Hind Abu Alamrain

Shiksha Kaharanthilake

October 24, 2025

**A blue and white background with squares

AI-generated content may be incorrect.**

**Student 1: Team Leader**

**Title : : Runtime Network Evidence Extractor**

**Student Name**: **Hind Abu Alamrain**

**Student ID: 300403234**

**Tool: Frida**

Frida is an open-source dynamic instrumentation toolkit that injects JavaScript into live processes, enabling developers and security analysts to observe, intercept, and modify an application’s runtime behavior without rebuilding it.

**Apps:**

* **Application 1:** Instagram
* **Application 2:** Telegram

**Novelty:**

This project introduces a runtime forensic approach that uses **Frida** with SSL-pinning bypass to expose and analyze network communications in **Instagram** and **Telegram**. It captures live network artifacts (tokens, headers, URLs) that are normally hidden, saves them as structured **JSONL** for later analysis and timeline generation, and delivers a repeatable set of Frida scripts and a report generator.

**Integrated Component:**

The two projects focus on complementary areas of mobile investigation—one applies **real-time network analysis and penetration testing** using Frida and SSL-pinning bypass, while the other performs **static and audio-forensic analysis** using Whisper and Pydub/FFmpeg. Together, they demonstrate how **dynamic runtime evidence** and **static forensic artifacts** can be analyzed side by side to provide a deeper understanding of user communication behaviors and app-level data flows across multiple mobile platforms. The integration will be achieved through a **joint automated report generation**, where both projects’ results are structured and summarized within one cohesive forensic output.

**AI Use Section:**

|  |  |  |
| --- | --- | --- |
| **AI Tool Name** | **Version, Account Type** | **Specific feature for which the AI tool was used** |
| ChatGPT (GPT-5 Thinking mini) | Plus | Frida – SSL pinning Bypass |

**Value Addition:**

* I read additional resources to better understand the topics and to develop this proposal
* I defined Frida in my own words based on my understanding from ChatGPT and other resources.

**Appendix:**

**Explain what is Frida, and what are its uses?**

**ChatGPT said:**

Frida is an open-source dynamic instrumentation toolkit. It lets you inject JavaScript into running processes (Android, iOS, Windows, macOS, Linux) so you can observe and modify what an app does at runtime—without rebuilding it.

What you can do with Frida

* Runtime analysis & debugging: Hook functions/methods, inspect arguments/returns, trace call stacks.
* Security testing: Probe protected code paths, test defenses (e.g., SSL-pinning bypass, anti-tamper/anti-debug), fuzz inputs.
* Reverse engineering: Explore obfuscated logic, discover hidden/undocumented APIs, extract in-memory strings/keys.
* Network forensics: Intercept or log HTTP/TLS calls to capture tokens, headers, URLs in real time (in authorized test setups).
* Malware analysis: Watch file, network, crypto, and persistence behaviors as they happen.
* Automation: Script repeatable hooks to validate hypotheses and generate reports/artifacts**.**

**What is the meaning of reverse engineering**

**ChatGPT said:**

Reverse engineering is the process of analyzing a finished system (software, hardware, or a protocol) to understand how it works internally—its structure, logic, data flows, and interfaces—usually when source code or design docs aren’t available.

What it’s used for

* Security: find vulnerabilities, verify patches, analyze malware.
* Interoperability: document file formats/APIs to build compatible tools.
* Forensics: reconstruct actions, timelines, and data handling in apps.
* Debugging/maintenance: understand legacy or third-party components.

Common approaches (software)

* Static analysis: study binaries/APKs without running them (e.g., Ghidra/IDA, jadx, apktool).
* Dynamic analysis: observe behavior while running (e.g., Frida hooks, debuggers, sandboxing).
* Hybrid: combine both to validate hypotheses.

Always ensure you have legal authorization—reverse engineering can be restricted by law, licenses, or terms of service.

**SSL-pinning bypass?**

**ChatGPT said:**

**SSL-pinning bypass** is the act of defeating an app’s client-side certificate/public-key pinning so the app will accept a different TLS certificate (for example, a proxy’s certificate). This allows you to intercept and inspect the app’s encrypted network traffic (headers, tokens, URLs) during testing or forensics — but only on apps/devices you own or are authorized to test.

**Quick breakdown**

* **SSL pinning:** app verifies the server’s certificate or public key against a known value baked into the app. If it doesn't match, the connection is rejected.
* **Bypass:** modify or neutralize the pinning check so the app treats a proxy’s certificate as valid and continues the TLS handshake.

**Why you might do it (legitimate reasons)**

* Debugging and troubleshooting network calls.
* Security testing / pentesting to validate server-side protections.
* Forensic collection of runtime artifacts in a controlled lab or with consent.

**Common bypass approaches (high-level)**

* **Hooking/patching the app’s pin-check functions** (e.g., okhttp3.CertificatePinner.check, X509TrustManager.checkServerTrusted) and return success.
* **Hooking platform APIs** (e.g., SecTrustEvaluate on iOS) to force trust.
* **Installing a custom root CA** on the device (when the app doesn’t pin).
* **Using dynamic instrumentation** tools like Frida to inject code at runtime and disable or override checks.
* **Patching the binary** (static modification) to remove or alter pinning logic.

**Risks & cautions**

* May expose sensitive user data — treat captured artifacts as highly sensitive.
* Illegal/unethical without permission; may violate ToS or laws.
* Can break app functionality if done incorrectly.

**Work log table:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Number of hours** | **Description of work** |
| 11-10-2025 | 4 | Tools searching |
| 12-10-2025 | 2 | Apps searching |
| 18-10-2025 | 3 | Choosing the tool & Apps |
| 22-10-2025 | 5 | Working on the proposal |
| 24-10-2025 | 1 | Working on the proposal |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Total Hours : 15**

**Closing and References:**

[**How to Detect Frida Toolkit Abuse in Your Mobile App**](https://fingerprint.com/blog/exploring-frida-dynamic-instrumentation-tool-kit/)

[**Welcome | Frida • A world-class dynamic instrumentation toolkit**](https://frida.re/docs/home/)

[**SSL Pinning Bypass: Essential Methods and Security Solutions**](https://www.startupdefense.io/cyberattacks/ssl-pinning-bypass)

[**Introduction to Frida for Reverse Engineering - DEV Community**](https://dev.to/phantomthreads/introduction-to-frida-for-reverse-engineering-1gc9)

**Student 2: Team member**

**Title : Voice and Audio Artifact Extractor**

**Student Name**: **Shiksha Kaharanthilake**

**Student ID: 300398278**

**Tool: Whisper & Pydub/FFmpeg**

These tools will be used for **audio transcription and conversion**. Whisper (by OpenAI) provides accurate speech-to-text capabilities, while Pydub or FFmpeg will handle **audio file format conversions** to ensure compatibility across various media sources.

**Applications:**

* **Application 1:** WhatsApp
* **Application 2:** Messenger

These communication platforms will serve as the **data sources**, providing real

world audio messages or voice notes for transcription, processing, and analysis.

**Novelty**

The proposed system introduces an innovative approach to **building schema extractors** and **automating artifact reporting** from conversational audio sources such as WhatsApp and Messenger. By integrating **Whisper** for transcription and **Pydub/FFmpeg** for audio conversion, the system enables automated extraction of structured information and generation of detailed reports. This approach improves **accuracy, efficiency, and scalability** in digital communication analysis and documentation.

**Integrated Component:**

The two projects focus on complementary areas of mobile investigation—one applies **real-time network analysis and penetration testing** using Frida and SSL-pinning bypass, while the other performs **static and audio-forensic analysis** using Whisper and Pydub/FFmpeg. Together, they demonstrate how **dynamic runtime evidence** and **static forensic artifacts** can be analyzed side by side to provide a deeper understanding of user communication behaviors and app-level data flows across multiple mobile platforms. The integration will be achieved through a **joint automated report generation**, where both projects’ results are structured and summarized within one cohesive forensic output.

**AI Use Section:**

|  |  |  |
| --- | --- | --- |
| **AI Tool Name** | **Version, Account Type** | **Specific feature for which the AI tool was used** |
| **ChatGPT** | **GPT-5 , Free account** | **How Whisper and Pydub/FFmpeg work together** |
|  |  | **How Audio and Voice collaborate in an audio processing + transcription pipeline**. |
|  |  |  |

**Value Addition:**

* Explored additional materials on audio processing, noise reduction, and speech recognition to make the project more effective**.**
* Used what I learned about Whisper, Pydub, and FFmpeg to explain how the Voice and Audio Artifact Extractor works.

**Appendix:**

* **Question of mine:** what is wisper and Pydub/FFmpeg?
* **Results given by ChatGPT:**

**1. Whisper**

* **What it is:** Whisper is an automatic speech recognition (ASR) system developed by

OpenAI. It can transcribe spoken audio into text.

* **Key features:**
* Supports multiple languages.
* Can handle noisy audio fairly well.
* Can also do translation of spoken audio in some versions.

**2. Pydub**

* **What it is:** Pydub is a **Python library for audio manipulation**.
* **Key features:**
* Load, save, and convert audio files between formats (.mp3, .wav, .ogg).
* Slice, merge, or modify audio (e.g., adjust volume, fade in/out).

**3. FFmpeg**

* **What it is:** FFmpeg is a **powerful multimedia framework** for handling audio and video. It’s **not Python-specific** but works behind the scenes for libraries like Pydub.
* **Key features:**
* Convert audio/video formats.
* Extract audio from video.
* Compress or modify media.
* **Relation to Pydub:**  
  Pydub relies on FFmpeg to read/write most audio formats. Without FFmpeg installed, Pydub cannot handle formats like MP3.

**In short:**

* **Whisper:** Speech-to-text AI.
* **Pydub:** Python library to manipulate audio files.
* **FFmpeg:** Backend tool that does the heavy lifting for audio/video processing.
* **Question of mine:** explain how wisper and Pydub/FFmpeg inetract with voice and audio
* **Results given by ChatGPT:**

**The Big Picture**

When you use **voice or audio** in a Python project (for example, a voice-to-text app), there are **three main stages**:

1. **Audio handling** — reading, cleaning, converting formats (that’s **Pydub + FFmpeg**)
2. **Audio analysis / AI processing** — extracting words, meaning, or other features (that’s **Whisper**)
3. **Output** — text, translation, or playback

**How they Interact**

**FFmpeg → Low-level audio engine**

* FFmpeg is the *foundation layer*.
* It’s a **multimedia processor** capable of decoding, encoding, and converting nearly every audio/video format.
* Pydub and Whisper both *depend* on FFmpeg to read raw audio data.

**Example:**  
When you load an MP3 file, it’s actually FFmpeg that:

* Opens the file,
* Decodes compressed audio (MP3 → PCM raw data),
* Provides audio frames to the next layer (Pydub or Whisper).

Think of FFmpeg as the **audio translator** that makes all formats understandable to Python.

**Paydub- High-level audio manipulation**

* Pydub uses FFmpeg under the hood.
* It gives you a **Pythonic interface** to work with audio — slicing, merging, amplifying, trimming, etc.
* It doesn’t understand “speech” — it just edits sound waves.

**So what it does:**

* Reads the audio file (via FFmpeg).
* Converts it into a manageable format (e.g., raw waveform data).
* Lets you modify it — e.g. make it louder, remove silence, change format.
* Saves the modified version, often as a .wav file for Whisper.

**Why this matters for Whisper:**  
AI models like Whisper need **clean, consistent, uncompressed** audio (WAV or FLAC).  
Pydub + FFmpeg prepare that input.

**Wisper- AI speech understanding**

* Whisper is an **AI model trained on millions of hours of speech**.
* It takes **raw audio waveforms** (not compressed MP3s) as input.
* Internally, it:
  1. Converts sound waves into **mel spectrograms** — a visual-like representation of sound frequencies.
  2. Feeds those spectrograms into a **transformer model**.
  3. Outputs **text tokens** — words it thinks were spoken.

**So Whisper’s job:**

* Understands the *voice content*.
* Transcribes (speech → text) or translates (speech → another language).
* **Question of mine:** what is mel spectrograms
* **Results given by ChatGPT:**

**What is a Spectrogram?**

A **spectrogram** is basically a **picture of sound** — it shows how the **energy (loudness)** of different **frequencies** changes **over time**.

* The **x-axis** = time ⏱️
* The **y-axis** = frequency (pitch)
* The **color/brightness** = intensity (how strong that frequency is)

So, if you say “hello,” the spectrogram shows how the sound waves of your voice rise and fall over milliseconds.

**What is a Mel Spectrogram?**

A **mel spectrogram** is a *special type* of spectrogram that’s **tuned to human hearing**.

The “mel” part comes from the **mel scale**, which spaces frequencies in a way that matches how humans actually perceive pitch — we hear **low frequencies** more distinctly than very **high ones**.

So, the mel spectrogram:

* Compresses high frequencies (where humans hear less detail)
* Expands low/mid frequencies (where we hear more differences)
* Gives a **human-like view** of sound patterns

**Why AI Models (like Whisper) Use It**

Whisper doesn’t process raw sound waves directly. Instead:

1. The raw waveform (a long line of numbers) → is **transformed into a mel spectrogram**.
2. That mel spectrogram (basically a 2D image) → is **fed into the neural network**.
3. The network then learns patterns in this image to recognize **words, accents, tones, and languages**.

**Work log table:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Number of hours** | **Description of work** |
| 12-10-2025 | **4** | Tools searching |
| 13-10-2025 | 3 | App searching |
| 15-10-2025 | 1 | Comparing tools and Apps found during the search |
| 18-10-2025 | 1 | Choosing the tool and Apps |
| 22-10-2025 | 5 | Working on the proposal |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Total Hours : 14**

**Closing and References:**

[**The voice-to-text AI that turns speech into clear, polished writing in every app.**](https://wisprflow.ai/)

[**How to Use Whisper AI A Step-by-Step Guide for Beginers**](https://aiwhisper.co.uk/how-to-use-whisper-ai-a-step-by-step-guide-for-beginners/)

[**How to use OpenAI whisper for speech recognition**](https://www.graphcore.ai/posts/how-to-use-openais-whisper-for-speech-recognition)

[**WhatsApp in Plain Sight: Where and How You Can Collect Forensic Artifacts**](https://www.group-ib.com/blog/whatsapp-forensic-artifacts/?)

**Project Contract:**

**Work Date/Hours logs for student:**

We, the undersigned, agree to the following working terms for this project:

| **Agreement Area** | **Description** |
| --- | --- |
| **Meeting Frequency** | Twice per week on Teams / In person |
| **Communication Method** | WhatsApp group, Email, GitHub comments |

**Team Signatures:**

| **Name** | **Signature** | **Date** |
| --- | --- | --- |
|  |  |  |
| Hind Alamrain |  | [24-10-2025] |

|  |  |  |
| --- | --- | --- |
| Shiksha Kaharanthilake | A black background with a black square  AI-generated content may be incorrect. | [24-10-2025] |