Android Evidence Forensic Database

Team: sdmay19-38

Members: Jacob Stair
Connor Kocolowski
Emmett Kozlowski
Mitchell Kerr
Matthew Lawlor

Clients/Advisors: Dr. Yong Guan, Dr. Neil Gong, Chris Cheng, Chen Shi

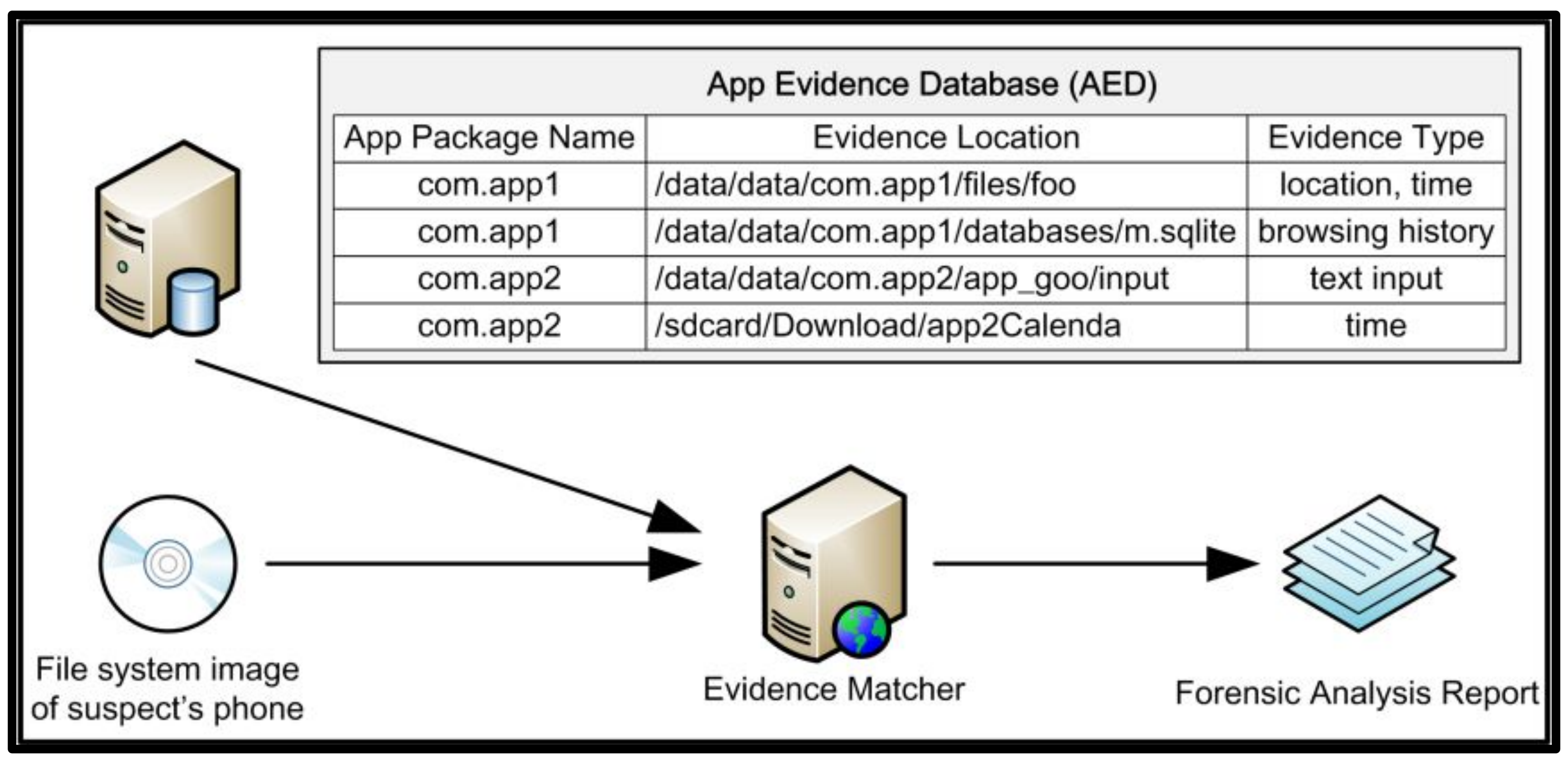
In Partnership with: The NIST Center of Excellence in Forensic Sciences - CSAFE at Iowa State University

Problem

Mobile device forensics has become a critical and high-demand necessity desired by law enforcement crime labs. With technology becoming more and more integrated into our lives, digital forensics has begun to play a larger role in proving the innocence or guilt of suspects. Every cell phone contains many mobile apps, which create records of all sorts of digital evidence such as GPS locations. Current digital forensic practices for finding this data are time consuming and error prone.

Solution

Our goal is to create a up-to-date real-world evidence database of Android apps from different app stores that are globally used. We will create web crawlers to traverse these app stores to collect metadata and download the application files. After collecting the file, it will be processed using the forensic analysis tools that collect where the application is storing the information that it gathers.


App Evidence Database (AED)
File system image of suspect's phone
Evidence Matcher
Forensic Analysis Report

App Package Name	Evidence Location	Evidence Type
com.app1	/data/data/com.app1/files/foo	location, time
com.app1	/data/data/com.app1/databases/m.sqlite	browsing history
com.app2	/data/data/com.app2/app_goo/input	text input
com.app2	/sdcard/Download/app2Calenda	time

Concept Sketch

Functional Requirements

- ➢ Crawler must collect all available apk files and all metadata that can be found on the webpage.
- ➢ Crawler must store metadata and apks into a database.
- ➢ Once collected, the apk will be passed through our client's forensic tool that will output information about where the app stores the data it collects.
- ➢ Database must store results from forensic program.
- ➢ Backend must allow adding new forensic reports to the database
- ➢ Frontend will allow users to query the database and display the results

Non-Functional Requirements

- ➢ Scalability - The system must be scalable to support the vast amount of applications we need to download and analyze.
- ➢ Availability - The system must be operating 24/7 to ensure that all versions are collected when they are updated on an app store.
- ➢ Maintainability - The system needs to be maintainable past the day that we deliver the product. Since webpages are constantly updated and changed the crawlers also need to be updated to support those changes.
- ➢ Data Integrity - The data that we collect from the websites and the forensic tool should not be modifiable by any individual after collection.

Operating Environment

We will need a physical server that can support the amount of memory space needed to run all the crawlers. The various web crawlers will be running simultaneously in containers on our server. We will also need a large storage space for all the information we will collect

Functional Modules

Web Crawlers

The web crawlers are used to collect the information and apk files for each application. Since each site has different html and functions differently, we created a crawler tailored to each store to collect data. The crawlers utilize Beautifulsoup to scrape each webpage and the Selenium driver for webpage interactions.

Backend

The Node.js backend uses the Express framework and is responsible for handling all the requests from the front end. The backend processes the request against the database and returns the desired information. It is also responsible for allowing the addition of new forensic reports to the database for an app.

Front End

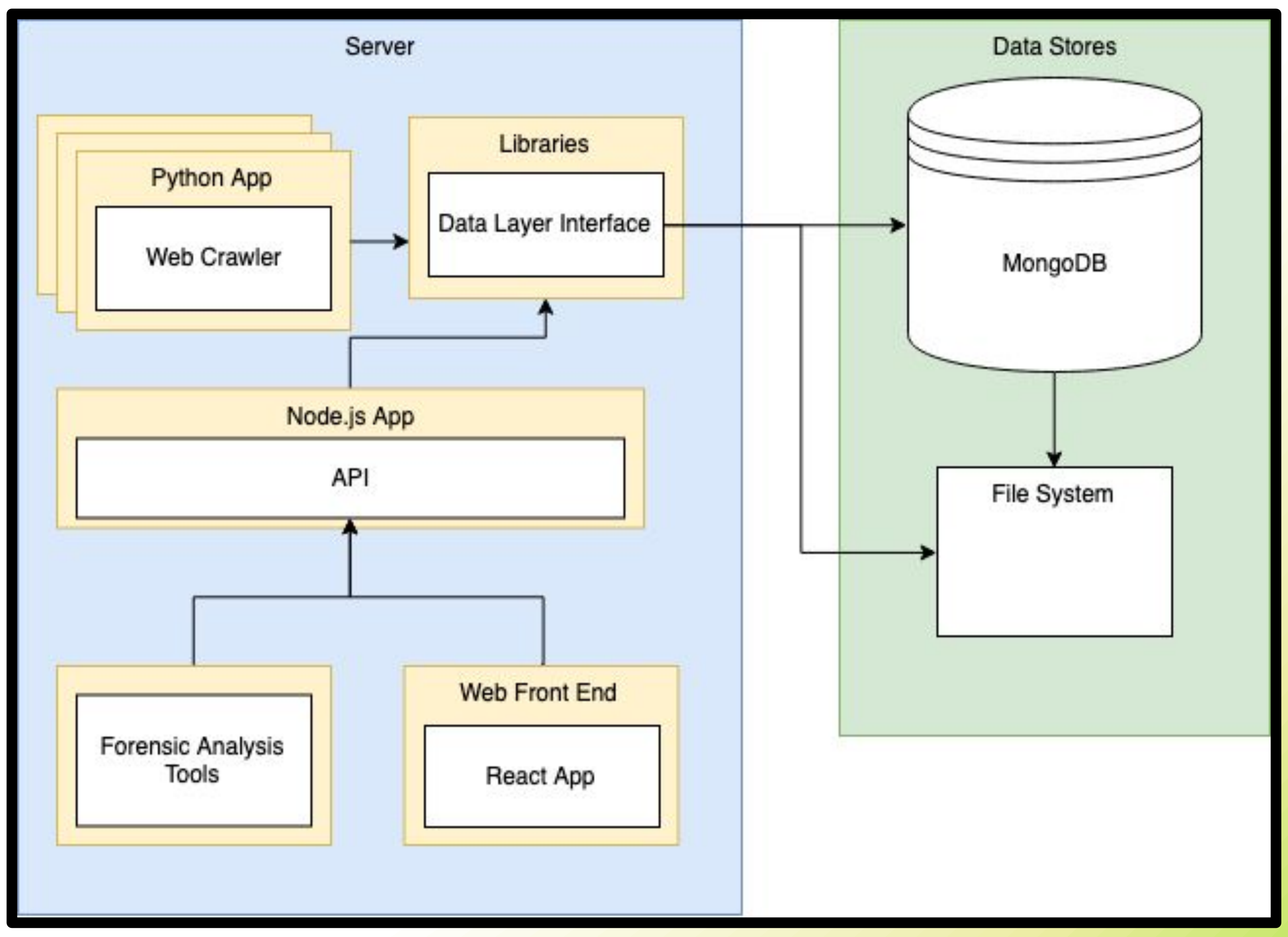
The frontend is a React app website. This website is the portal for users to interact with our database. It allows users to query information for a particular app or a list of apps. It will then display the corresponding information in an easy to understand format.

Data Stores

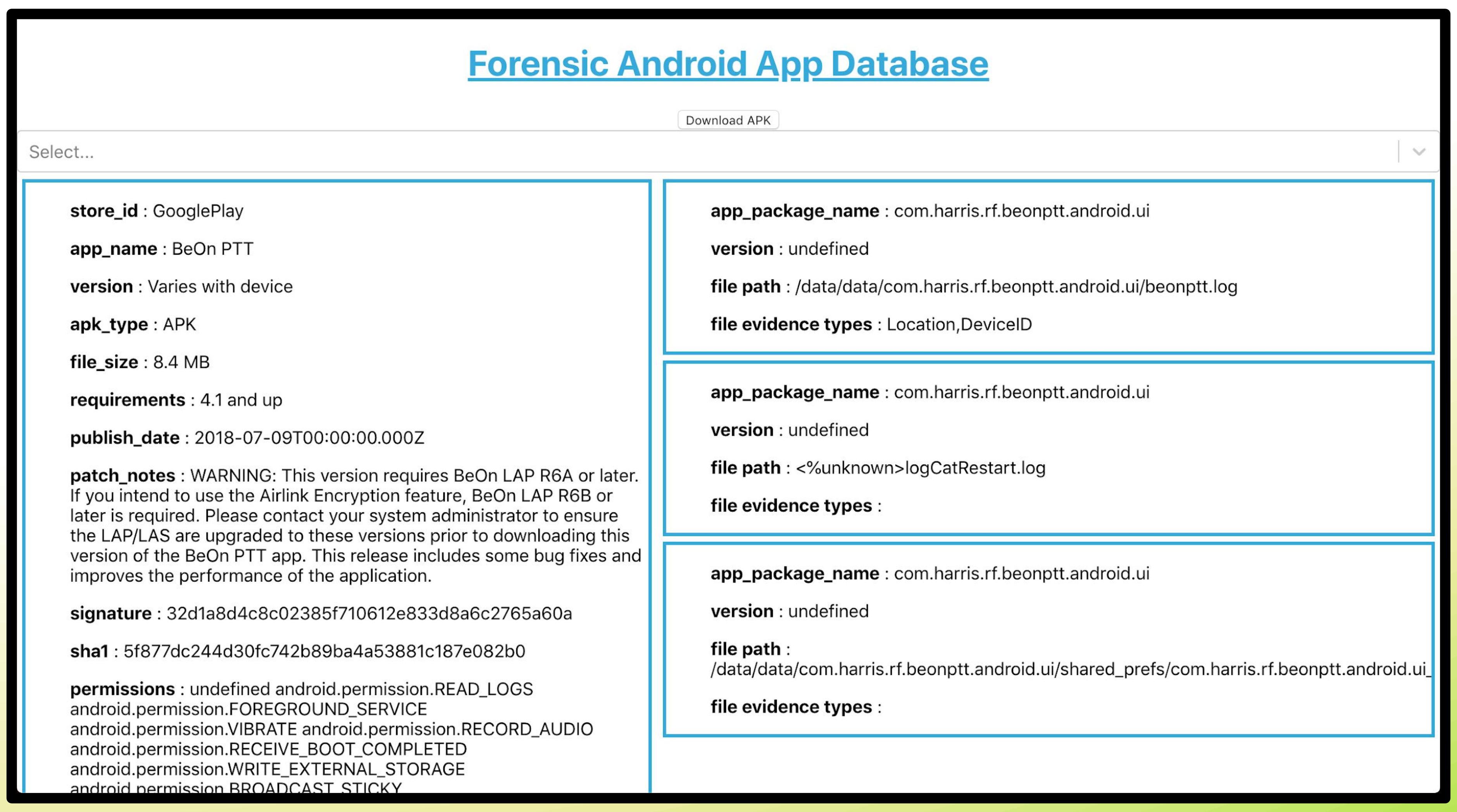
We have two main data stores for our project. The information about an application that is collected off of a store webpage gets stored into a MongoDB instance. The actual apk files of the applications are stored in a separate file system and the path to the file is linked to the app entry in MongoDB.

Intended Users and Use Cases

- ➢ The intended end user for our database will be digital forensic investigators. One such situation it might be used in is locating the useful data from an application that might be stored on a device. Investigators and their teams will be able to use our web interface and search for any and all applications which are present on the device to see where relevant information is stored.
- ➢ Another possible user of our database will be academic researchers. Researchers whose studies relate to mobile applications and their contents may find it beneficial to have the ability to search through many of the available applications across various third party app stores.



System Diagram



store_id : GooglePlay

app_name : BeOn PTT

version : Varies with device

apk_type : APK

file_size : 8.4 MB

requirements : 4.1 and up

publish_date : 2018-07-09T00:00:00.000Z

patch_notes : WARNING: This version requires BeOn LAP R6A or later. If you intend to use the Airlink Encryption feature, BeOn LAP R6B or later is required. Please contact your system administrator to ensure the LAP/LAS are upgraded to these versions prior to downloading this version of the BeOn PTT app. This release includes some bug fixes and improves the performance of the application.

signature : 32d1a8d4c8c02385f710612e833d8a6c2765a60a

sha1 : 5f877dc244d30fc742b89ba4a53881c187e082b0

permissions : undefined android.permission.READ_LOGS android.permission.FOREGROUND_SERVICE android.permission.VIBRATE android.permission.RECORD_AUDIO android.permission.RECEIVE_BOOT_COMPLETED android.permission.WRITE_EXTERNAL_STORAGE android.permission.BROADCAST_STICKY

app_package_name : com.harris.rf.beonptt.android.ui

version : undefined

file path : /data/data/com.harris.rf.beonptt.android.ui/beonptt.log

file evidence types : Location,DeviceID

app_package_name : com.harris.rf.beonptt.android.ui

version : undefined

file path : <%unknown>logCatRestart.log

file evidence types :

app_package_name : com.harris.rf.beonptt.android.ui

version : undefined

file path : /data/data/com.harris.rf.beonptt.android.ui/shared_prefs/com.harris.rf.beonptt.android.ui_

file evidence types :

Web UI

UI Description

The photo on the left is a screen shot from our website. In this particular instance we can see the detailed information about the selected app, BeOn PTT. The left side contains relevant metadata about the application. The right side contains all the report data associated with the selected version of that application.

Future Plans

- ➢ Add more crawlers to include more stores
- ➢ Add additional features to the UI to improve the user experience
- ➢ Move to a more permanent setup for external users.

Testing

- ➢ The web crawlers have automated unit tests that verify accurate data scraping from web pages
- ➢ We have manual testing to ensure that the crawlers are collecting all the information from the websites correctly.
- ➢ The backend has integration tests using mocha and chai for each endpoint.
- ➢ The frontend is using a testing framework called Jest to mock our javascript functions.
- ➢ The frontend also implements manual testing to check visuals.