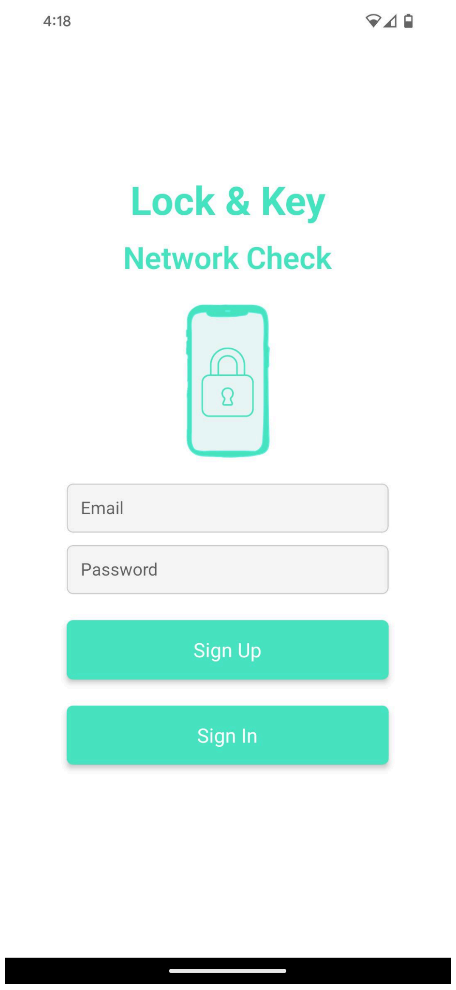
Riley Schina
William Zhongjun

Lock & Key: Network Check

Group 2 Final Report



Abstract

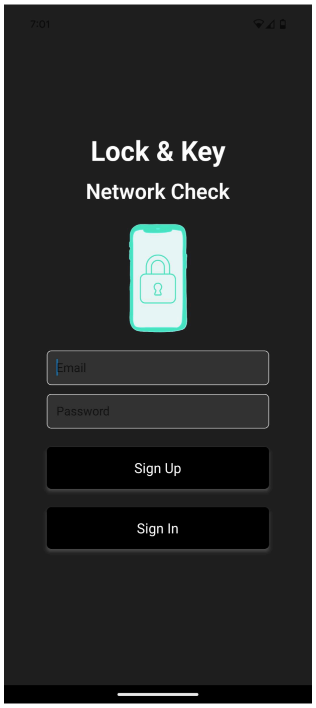
Lock & Key: Network Check is an app focused on network and device security in an increasingly connected world. Making use of tools like nmap scanning, geolocation, and vulnerability databases to help people make an informed decision on their security.

The app searches for local machines on your network and helps to identify vulnerabilities within the system, then directs the user to resources to help. By focusing on making it easier for people to check their own networks and devices and find relevant information to help meet their network security needs.

Completion Report

User Authentication

For this project, Firebase authentication was used as an efficient and secure login service. Email and password verification and registration was chosen as it is the most commonly used method for secure login. For the purpose of testing, the app has a setup account using **test@test.com** as the login, and **123456** as the password.



Data Storage

The app uses two main types of storage: Asynchronous storage and Firestore databases, for local, simple data, and database, complex data respectively. The Async data is used to control the dark and light mode for the app, a boolean is stored locally on the app so that it remembers between uses which style the user prefers. The Firestore database is storing the devices found in our scans, a log of previous scans, the CVE information for those scans, and the geolocation data of the networked devices. This information is unique to each user, with the database structure being **User > Log >Device> Device Information and CVE Information**.

API Integration

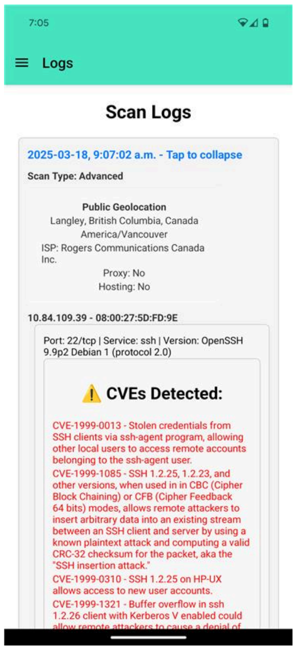
- nmap API:(my own api)
    - Used to perform network scans.

    - Fetches open ports, running services, and potential vulnerabilities.

- IP Geolocation API (e.g., ipapi.co):
    - Retrieves geolocation data of detected IPs.
    - Cross-checks this data with device GPS to flag mismatches (potential proxy/VPN use).
- Axios API
    - Acts as a bridge connection between the device and the CVE database
    - Uses HTTPS GET method to make the link
- CVE Database API:
    - Maps detected services/ports to known vulnerabilities.
    - Provides actionable insights to users by identifying potential risks.
- Firebase API (Authentication & Storage):
    - Manages user authentication (email/password login).
    - Stores scan logs and sensitive data securely.

UI and Navigation

The UI for the app has a focus on simplicity and legibility. The app contains a large amount of Flatlist items that need to be easy to read, so the containers for each section of data are slightly different colours and are outlined. Standardizing the sizes of buttons, text, titles, and more was done to also enhance legibility. As recommended, a dark and light mode was implemented to allow users to choose their preferred styling for the app.


7:05

Logs

Scan Logs

2025-03-18, 9:07:02 a.m. - Tap to collapse

Scan Type: Advanced

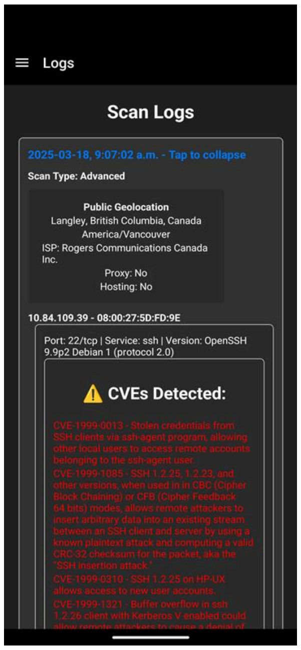
Public Geolocation
Langley, British Columbia, Canada
America/Vancouver
ISP: Rogers Communications Canada Inc.
Proxy: No
Hosting: No

10.84.109.39 - 08:00:27:5D:FD:9E

Port: 22/tcp | Service: ssh | Version: OpenSSH 9.9p2 Debian 1 (protocol 2.0)

⚠️ CVEs Detected:

CVE-1999-0013 - Stolen credentials from SSH clients via ssh-agent program, allowing other local users to access remote accounts belonging to the ssh-agent user.
CVE-1999-1085 - SSH 1.2.25, 1.2.23, and other versions, when used in in CBC (Cipher Block Chaining) or CFB (Cipher Feedback 64 bits) modes, allows remote attackers to insert arbitrary data into an existing stream between an SSH client and server by using a known plaintext attack and computing a valid CRC-32 checksum for the packet, aka the "SSH insertion attack."
CVE-1999-0310 - SSH 1.2.25 on HP-UX allows access to new user accounts.
CVE-1999-1321 - Buffer overflow in ssh 1.2.26 client with Kerberos V enabled could allow remote attackers to cause a denial of


Logs

Scan Logs

2025-03-18, 9:07:02 a.m. - Tap to collapse

Scan Type: Advanced

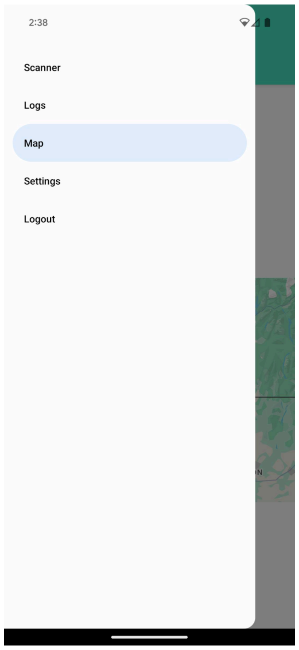
Public Geolocation
Langley, British Columbia, Canada
America/Vancouver
ISP: Rogers Communications Canada Inc.
Proxy: No
Hosting: No

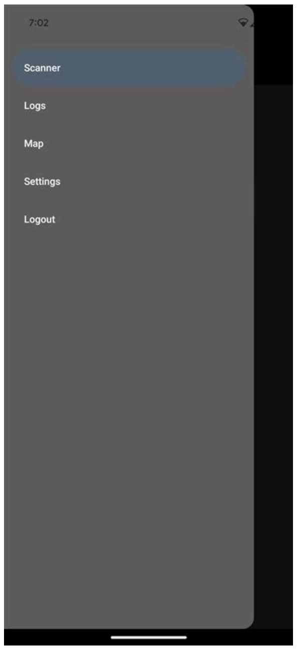
10.84.109.39 - 08:00:27:5D:FD:9E

Port: 22/tcp | Service: ssh | Version: OpenSSH 9.9p2 Debian 1 (protocol 2.0)

⚠️ CVEs Detected:

CVE-1999-0013 - Stolen credentials from SSH clients via ssh-agent program, allowing other local users to access remote accounts belonging to the ssh-agent user.
CVE-1999-1085 - SSH 1.2.25, 1.2.23, and other versions, when used in in CBC (Cipher Block Chaining) or CFB (Cipher Feedback 64 bits) modes, allows remote attackers to insert arbitrary data into an existing stream between an SSH client and server by using a valid known plaintext attack and computing a valid CRC-32 checksum for the packet, aka the "SSH insertion attack."
CVE-1999-0310 - SSH 1.2.25 on HP-UX allows access to new user accounts.
CVE-1999-1321 - Buffer overflow in ssh 1.2.26 client with Kerberos V enabled could allow remote attackers to cause a denial of





The navigation chosen for the app was Drawer navigation, this was chosen due to users needing to jump between different unrelated screens, and having a unified menu for changing screens made that easier. The dark and light mode was applied to the banners and the navigation of the app as well.

Native Features

The native feature chosen for this application is vibration. Once the scan is complete, the device vibrates as a way to stealthily inform the user of the completion of scans to deter shoulder surfing security guards in LAN penetration testing environments

App Functionality

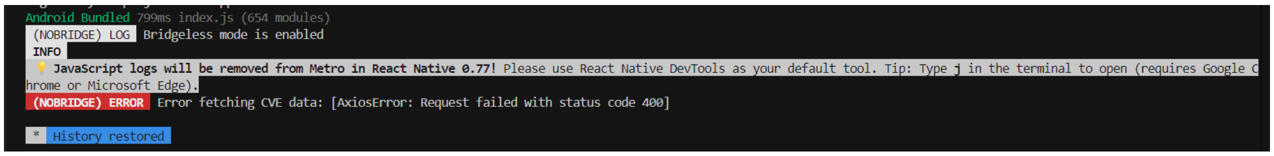
Button to initiate nmap scan (netword scanning), toggleable options between normal nmap -sV (service detection) or advanced nmap -A (all detectable network signatures), it also calls IP-geolcoation APIs and maps it on google maps. Logs are stored in Firestore, and authentication is through firebase auth. The app also has dark/light mode and some metasploit framework functionality for quick footholds.

Unachieved Functionality

True native network scanning triggered and executed by the mobile phone itself is impossible with React Native and out of the scope of this course.

Challenges

One of the major stumbling blocks for the project was the implementation of Axios to access CVE (Common Vulnerabilities and Exposures) databases. While Axios is a simple API to use, it has comparatively poor error catching and troubleshooting methods. Similarly the database that was used for the project came from an American government department, NIST, having somewhat outdated information for how to connect to and parse the data.



```
Android Bundled 799ms index.js (654 modules)
 (NOBRIDGE) LOG  Bridgeless mode is enabled
 INFO
 💡 JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (requires Google C
hrome or Microsoft Edge).
 (NOBRIDGE) ERROR  Error fetching CVE data: [AxiosError: Request failed with status code 400]

 *  History restored
```

The solution to this problem was a large amount of catching errors using console log messages, and using a brute force approach to finding what and where things went wrong. Console log messages were added to each segment of code until the issues could be properly identified. Parsing the CVE information was solved by carefully reading through the JSON of a specific search term to identify the proper calls to make in the code for relevant information.

As mentioned above, nmap needs a net card of the physical device to actually work, which is locked behind android and IOS phones unless rooted/jailbroken, and react native isn't actually native, so an intermediary solution is required.
Given the target audience as somewhat advanced penetration testers, it is assumed that they are likely to have a Kali Linux machine around. Hence, the locked backend feature is pushed to the laptop via ssh instead. This doesn't really affect the main goal of the application as a stealthier version of LAN scanning as the user can either carry the laptop via a backpack or leave it somewhere, which is considerably less suspicious compared to a stranger walking around with a laptop.

SSH with react-native-ssh doesn't work with expo's simulated js environment and doesn't work on IOS systems outright, so the solution is to have a node.js backend that is hosted on the laptop or ported over to the mobile device itself via Termux. And the APP itself just treats the backend as any other API request.

Reflection: Riley Schina

This project was a challenge for me, I had never used javascript before this course, so coding in React-Native, a javascript like language was unfamiliar and took quite a learning curve. Additionally, I had never dealt with security based coding and practices, with most of my experience in the subject being the use of anti-virus and firewall software; there was a great deal of terms, practices, and general knowledge that were new to me. A large amount of my time for this course and project were spent trying to learn what I could of the coding while reading up on practices for internet security. While the coding language is not my favourite, I learnt quite a bit of proper implementation within react native and for mobile development and have progressed my knowledge for security and networking

Reflection: Qian Zhongjun

Having just completed CompTIA Security+, this project served as a golden opportunity to apply that knowledge in a practical, mobile-first context. I approached it from a red team perspective, focusing on LAN reconnaissance and CVE detection via nmap and CVE APIs. While React Native and JavaScript were unfamiliar territory, the experience pushed me to adapt quickly—especially in managing asynchronous data flows, UI state, and API integration within the constraints of Expo. Emulators were at times difficult to configure, but I eventually developed a functional scanner UI, complete with Firebase authentication, native vibration feedback, and backend interaction over SSH. I also made deliberate design decisions to maintain stealth and operational realism—such as opting for vibration over push notifications to reduce the risk of exposing scan activity to shoulder-surfers or guards during a physical red team engagement.