Tell us what your idea is.

Everyday we encounter numerous web links across various apps ranging from social media to email, most of which are shortened URLs and hyper-linked texts, making it impossible to know where they might lead to. Some malicious web links are even disguised with legitimate website names to trick the average user. The internet is indeed a boon. But it is fair to say that for the average user, the internet is a black box and bad actors take advantage of this, whether it be stealing personal information, mining crypto-currencies or promoting illicit content.

A safe web experience should begin even before the user visits a website. When the user tries to open a suspicious weblink on any app, FalseLink intercepts to verify the content and credibility of the website it leads to and informs the user before he opens it in a browser.

To achieve this, FalseLink uses 2 stages of verification:

1. Malicious URL detection - Detect phishing URLs and repeated redirecting URLs.
2. Web content classification - Extract and determine the content of a website using Text/Image classification. (adult content, illegal betting websites, advertising spams etc.)

Existing solutions either requires the user to continually share their browsing information with 3rd parties or simply black-lists websites that are considered to be known bad-actors. Leveraging on-device machine learning enables us to perform low-latency-real-time verification on the device without ever having to share the user's browsing information.

Demo and Project Overview : https://www.youtube.com/watch?v=EfTHhnq321E

Tell us how you plan on bringing it to life.

Android Implementation Details

One of the most crucial aspects of my proposal is providing a seamless experience to the user. The user shouldn't have to copy a web link from one app, paste it onto another app to verify it and then open it in a browser! Thanks to the open nature of Android, this is not the case. The verification can be done as soon as the user clicks on a link and can be automatically redirected to a browser through the following steps:

1. Intercept web links by registering an intent-filter for web-URL schemes. Show the verification dialog on top of the host app as soon as the link is clicked. (does not require background services or any special permissions)
2. Extract the web content using jsoup HTML parser. Perform verification in the 2 stages as mentioned above.
3. Inform the user the results of the verification on the same dialog and pass the web link intent to the default browser if it is safe/white-listed or if the user explicitly chooses to visit it.

Sample code : https://github.com/chandruscm/FalseLink

- Android-specific implementation of the project is pretty much complete. Ready to train and plug-in tensorflow-lite models. The sample code shows a mock verification result when web-links are clicked.

Listed below are three ways to implement this proposal. For the intents and purposes of this challenge, I have chosen to go with the first method just for the ease of prototyping.

a) Stand alone app

Act as an intermediary, intercept web links, perform verification and pass the web link to the default browser.

Pros

- Compatible with all browsers.

Cons

- Cannot intercept malicious web links that are opened within a browser.

b) Browser-level implementation

Pros

- Verify links that are clicked even within the browser.

Cons

- Incompatibility with other browsers.

c) WebView integration

- This would be the ideal way to provide a system level protection for Android. I think FalseLink is a good proof-of-concept to evaluate whether an on-device ML-based solution is viable.

On-Device Machine Learning

I plan to implement the verification using TensorFlowLite models that can be deployed on-device using MLKit.

1. Malicious URL detection : Train a model to detect malicious URLs using URL tokens and information from whois entries.
2. Web content classification - Train models to classify text/images in a website to classes such as adult, betting-gambling, violence, spam, phishing, illegal. Jsoup parser is used to extract text and image features.

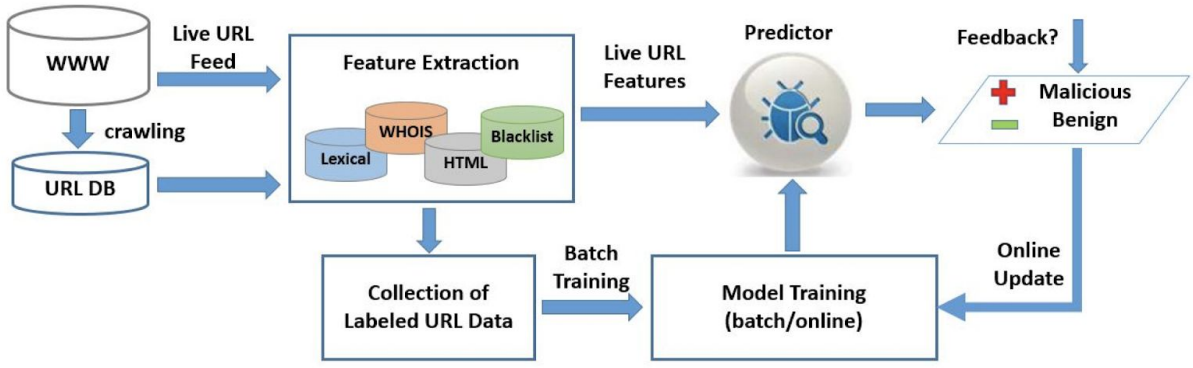

Fig. 2. A general processing framework for Malicious URL Detection using Machine Learning

Reference : This recently published paper highlights most of the research that has been done in malicious URL detection using Machine Learning. Interestingly, an on-device machine learning approach in the context of a mobile device has not been researched yet.

Support from TU Munich

The timing of this challenge couldn't have been more perfect! I am currently a master's student at the Technical University of Munich. In my current semester I am taking part in a course called Applied Machine

Learning, in which a set of 12 students including myself, have to apply machine learning techniques in a real-world use-case. My course instructor is happy to let me work on this proposal which means that I would receive technical support and expertise from my university!

How Google can help?

Basically connect me with my favorite Google developer advocates 😊

- Video session with Nick Butcher(@crafty) to review the UI/UX of the app.
- Video session with Florina Muntescue (@FMuntenescu) for code-reviews.
- Video session with a technical representative from the Android WebView team and Google SafeBrowsing team to discuss potential solutions for detecting malicious URLs and websites.
- Vaibhavi Desai (Googler) as a mentor for my project, who is currently running Machine Learning Bootcamps. She has previously mentored me for a Google student program called Applied CS with Android back in India.

Proposed Timeline

November

- ✔️ Setup project architecture (kotlin, MVVM, dagger2, room, coroutines, databinding).
- ✔️ Implement a PoC for the user-flow, intercept web links and pass it to browser.
- ✔️ Finalize the UI/UX of the app.
- ✔️ Setup verification pipeline with jsoup, room and tensorflow.

December - mid January

- 🔵 UI/UX review with Nick Butcher.
- 🔵 Research machine learning solutions. (Video session with WebView and SafeBrowsing team)
- 🔵 Implement stage 1 URL verification.

mid January - mid March

- 🔵 Code review-1 with Florina Muntescue.
- 🔵 Implement stage 2 web content classification.
- 🔵 Benchmark results.
- 🔵 UI design and improvements.

mid March - April

- Code review-2 with Florina Muntescue.
- Write tests.
- Prepare release build.
- Prepare store listing material for Google Play.
- Begin brief alpha-testing.

Tell us about you.

I'm Chandramohan Sudar, a master's student at the Technical University of Munich where I specialize in Machine Learning and Analytics. I have been working with Android for the past 5 years on projects ranging from Network Security to Object detection for Autonomous vehicles. Along with my master's study program, I work as a part-time Android developer at P3 digital services where I work on Android Automotive solutions for one of the major auto manufacturers in Germany.

I have also previously worked with **on-device machine learning** during an entrepreneurial course at my university where we partnered with Huawei to harness the on-device machine learning capabilities of the new Huawei smartphones running the Kirin 970/980 processors making use of their HiAi engine. My team and I developed Regalo, an AI-based Android application that recommends gifts based on the preferences of the recipient gathered from social media profiles. View the marketing video of the app here. View the demo of the app here.

During my bachelor's in India, I was part of Google's student program **Applied CS with Android** through which I delivered Android workshops to students in my university. I was one of 2 students from this program to receive a travel grant for **Google I/O 2016**. Looking back, the experiences I gained through the program and Google I/O, kick-started my journey into proper Android development and my involvement with the community through Google Developer Groups and other meetups. Blog post about my experience.

I also independently develop and publish Android apps in my spare time.

Orble - https://play.google.com/store/apps/details?id=com.chandruscm.orble.android

A minimalist reflex-based game developed using LibGDx, a cross platform game development framework. It has grossed more than 150,000 downloads with over 1000 reviews. It has also been featured in a lot of youtube channels.

Thirukkural Puthiya Vadivam (Tamil | English)

https://play.google.com/store/apps/details?id=com.chandruscm.thirukkural

An intuitive and reliable digital version of a 2000-year old Tamil literature classic.

Having noticed the lack of good quality Android apps in my mother tongue Tamil, I began developing this app with the core intention of developing a Tamil app that set a standard. Interestingly, I faced very specific challenges pertaining to Indian regional languages.

I have made use of the newest technologies and the best practices conforming to what is now termed as "Modern Android Development". (Kotlin, Dependency Injection with Dagger, Clean MVVM Architecture, Data binding, Jetpack Architecture Components, new Material Library). Due to the commercial aspect of the app and the business relations with a 3rd party publisher, I cannot open-source the app. However, if the committee wants to view this project source, I can definitely provide access to the GitHub repo.

miniRTO - https://play.google.com/store/apps/details?id=com.chandruscm.minirto

An app that fetches the registration details of Indian vehicles using jsoup Java HTML parser. It grossed more than 50k downloads with a rating above 4.0. Unfortunately I had to discontinue the app due to govt. regulations, but this ended up making the users unhappy which resulted in the current 2.8 rating 😐

- 💡 Blog - https://chandruscm.wordpress.com
- 💡 Medium - https://medium.com/@chandruscm
- 💡 GitHub - https://github.com/chandruscm
- 💡 LinkedIn - https://www.linkedin.com/in/chandruscm/
- 💡 Portfolio - http://chandruscm.wordpress.com/portfolio/