Group 36 Project

# Security Analysis in Mobile Web Browsers

Group Members: Isaiah Genis(ig596@nyu.edu), Kris Acevedo (ka2765@nyu.edu), Nina Hartney (nh1790@nyu.edu), Savas Konstadinidis(sk9176@nyu.edu)

## Abstract

As our daily lives become more mobile-driven, we have often removed the need for a separate desktop/laptop for computing and the shift to mobile exposes users to an increased threat footprint. Several notable mobile device vulnerabilities include weak encryption and outdated applications/software. While many mobile applications appear similar to their website, their core components are vastly different; one considerable concern is which scenarios will prompt a user to allow access to a resource versus those that will not. Our project analyzes how the security of mobile browsers and their privilege structure can be exploited in ways a malicious web app or traditional browser likely could not.

## Section I: Introduction

The popularity and widespread adoption of smartphones over the last decade has considerably spurred the rise of mobile applications (referred to as apps hereafter). Android has become the largest share of the smartphone market worldwide, with mobile apps penetrating nearly all industries, such as finance, healthcare, education, and transportation as apps have become increasingly significant in our daily lives (Jog et al., 2017). According to Statista, as of June 2022, there are over 2.6 million different apps accessible on Google Play, the world's largest Android app store. The vast majority of mobile users keep sensitive and private information on their smartphones ranging from confidential pictures to bank accounts and crypto coins. Android portable devices are the prime target for criminals who are seeking to access and steal the private data the users carry in their devices. A common way to achieve that is by publishing rogue software (Huang et al., 2015).

One of the primary security concerns is the permission procedure (Zhang et al., 2016). Although Android's permission mechanism requires programs to obtain permission before accessing sensitive user data, such as contacts and SMSs, or specific system services, such as camera and Internet access, the permission mechanism does not protect from device reconnaissance or other web vulnerabilities via a website in a mobile browser. These include items such as enumerating the various input devices (microphones, cameras, etc.) or Cross-origin resource sharing (CORS) requests. iOS-based browsers are implemented with the same Safari WebKit browser engine configuration, Android, on the other hand, utilizes a broad variety of browser engine implementations.

The organization of this paper is as follows. Section II outlines the background and current methods of implementation. Section III will describe the motivating example of why this deserves investigation. In Section IV, we will explain our hypothesis. Section V analyzes the relevant concerns, and we conclude and discuss future work in Section VI.

## Section II: Background

For this paper, we should consider Android's shift toward advanced privacy features, as the previous operating system (OS) updates have included significant security improvements regarding privacy. One of Android's most prominent enhancements is the permissions process and how permissions are requested. Beginning in Android 6.0 (Marshmallow), each app must request for the permission they require. It is the developer's responsibility to ask for permission during the app's runtime, compared to previous versions where permissions were granted by default. However, in many cases, the permissions that are being requested appear to be unnecessary for the performance and usability of the app.

Each Android app runs in a limited-access sandbox, if an app needs to access resources or information outside of its own sandbox, the app must request the appropriate permission. Common permissions an app generally requests:

- *Storage (Read and/or Write)*
- *Location Access*
- *Contacts (Read and/or Write)*
- *Phone Calls*
- *Camera*
- *Network Communication*

Researchers have created user-oriented permission prompts to mitigate the issues with permissions and permissions procedures to ensure that smartphone users are appropriately advised of the rights required by apps. However, due to the complexity of Android's permission structure, most of these efforts have proven futile (Wijesekera et al., 2015; Acar et al., 2016).

## Section III: Motivating Example

The expanded capabilities of web content loaded in a mobile web app is a crucial distinction between mobile and a normal web app. Recently, standard web browsers have started to make Application Programming Interface (API), such as location services, available to web applications. However, mobile web apps can combine standard web application capabilities with the functionality offered by mobile apps. With this combination, application developers and programmers can produce complex applications that are difficult to replicate in standard browsers.

Mobile Web Applications are typically developed using JavaScript or CSS, where the browser is used as the runtime environment, making the web application flexible for all mobile operating systems as (Jog et al., 2017). In this type of development, the app is built similar to a website, which is interpreted and rendered by the mobile browser (Jog et al., 2017). Mobile web applications function the same as traditional web applications, where a web server is in the cloud and is accessed through a web browser like Chrome or Safari (Jog et al., 2017).

Initially, mobile web apps did not have permission to use the device's hardware resources such as a camera or microphone; unfortunately, as smartphone technology advanced, these permission expanded and evolved as mobile browsers can optimize websites and the browser standardizes the web application, which allows the website to function smoothly on different devices (Jog et al., 2017)..

In previous years, attacks on mobile devices were primarily focused on social engineering attacks such as sending text messages or calling user phone numbers to scam users out of money (Boksasp & Utnes, 2012).

Today, mobile attacks have migrated towards stealing the private data contained on the devices with little user interference or knowledge (Boksasp & Utnes, 2012). These attacks often come from malicious apps in the Google Play store, however, the mobile browser and malicious websites are often overlooked attack vectors.

A significant enabler is that most users do not completely comprehend Android's permissions structure. Mobile users frequently ignore the permissions prompts and approve apps' requests for permissions without inspecting the instructions (Felt et al., 2012b). As a result, apps can simply obtain additional permissions, increasing the danger of user privacy leaks. This is the excessive privilege problem (Felt et al., 2011a). The security of Android is significantly reliant on the effectiveness of this permission mechanism. A critical risk is that a rogue program or site may secretly seek extra permissions. Such data is vulnerable to malicious activity, which has become a significant vulnerability of the entire mobile ecosystem.

## Section IV: Hypothesis

This permissions model discussed in the previous sections is further complicated when considering the websites that request to interact with system resources through web browser applications. This adds a new attack vector to requesting and accessing resources. With each site requesting access to several resources such as location, payment data, and contact details becoming more common, users are more likely to feel fatigued and give less scrutiny to the permission requests. Even the authors of this paper have had to allow cross-site tracking and cookies in iOS safari to use Duo and login to NYU from their phones in Safari or the NYU App (Duo, n.d.). Additionally, users assume that there is an equivalent level of browser-based security for mobile as for the desktop, which may be a dangerous assumption as we will establish.

This paper seeks to explore exactly what the boundaries of the security and permissions of browsers are and see what actions can be performed through them to collect information from or otherwise exploit users. For example, we take it for granted that sites will pivot to landscape when we turn our phones, but how do they know how we turn our phones without asking? With the number of traditional web vulnerabilities being possible on Mobile devices, there are likely opportunities for privilege escalation and other exploitation.

## Section V: Analysis

There have been numerous vulnerabilities found in the Web Applications of Mobile Devices that can lead to exploitation and privilege escalation. Rafay Baloch is a security researcher who has demonstrated multiple times that the mobile browsers used every day by millions of android users are not as safe as they believe. At the 2016 BlackHat Asia conference, Baloch highlighted several cross origins and cross scheme exploits that would allow for credential harvesting, remote code execution, mixed content bypass, and more (Baloch, 2016). Each of these exploits were directly related to insecure browser implementations, as Baloch identified that most Android mobile web browsers did not support the Content-Security-Policy Header, a standard part of the defenses adopted in traditional desktop web browsers (Baloch, 2016). In October 2021 there was a full chain exploit from a Samsung phone utilizing Android's browser. The following exploits, CVE-2021-38003 and CVE-2021-1048 enabled the attackers to infiltrate the browser and delivered a one-time link via email that spoofed URL shortening services. Once clicked, these URLs directed the victims to an attacker-owned domain that delivered Alien, Android malware that loads the Predator spyware and performs operations for it.The vulnerabilities discussed caused a significant impact on the connection of Android's mobile devices (Hardcastle, 2022).

In 2016, 18-year-old Meetkumar Hiteshbhai Desai, identified an iOS WebView bug that allowed auto-dialing of an attacker-controlled number and at the same time, controlled the user interface (UI) blocking the phone's user from canceling the call (Cimpanu, 2016). This vulnerability resulted in a denial-of-service attack on 911 services and Desai’s arrest (Cimpanu, 2016). Additionally, the vulnerability was found in the Twitter and Linkedin Applications, which are simple WebView wrappers, as such a vulnerability turns every website and tweet into a potential critical infrastructure attack (Cimpanu, 2016).

An analysis of Safari uncovered several interesting findings for the iOS based browser. It appears most modern desktop browsers will trigger an alert for uploading or downloading malicious files through Safe Browsing configurations, however this capability is not configured on Mobile devices. An important concern is to automatically open any URL (even in documents such as pdfs) with a single click. Without inspection of the URL by the user first, this could be an embedded malicious URL. In 2020 Baloch published an additional set of vulnerabilities related to Mobile Browser Security, which focused on Address Bar Spoofing vulnerabilities. Address Bar Spoofing vulnerabilities allow the attacker to display a fake destination address for the page (Baloch, 2020). By leveraging Cross-Site Scripting to spoof the address bar and pose as other sites on secure channels, the attacker can remove much of the TLS and domain-level security checks that would generally pose the most challenging part of a phishing site (Baloch, 2020).  The implications of this are that users believe they are logging in to Amazon or Apple when they provided their credentials to an attacker or have been granted additional device-level access to a malicious site.

## Conclusion & Future Work

The permissions scheme for mobile devices is overly complex for most users, and there is an ever-evolving maturity concerning web security for mobile devices. App permissions are playing an increasingly significant role in securing the device and data from misuse. It is vital for the user to keep examining if each app legitimately requires access to all parts of the phone, especially the more personal data and sensitive information user’s store on a device. Even for users that try to administer their permissions effectively, there are web exploits that will allow consent to the wrong party unintentionally, defeating the purpose of the site-based permissions.

In Safari, many of these settings are not controlled at a per-site level. Most users lack the technological sophistication to know how to effectively manage cross-site settings. The attempt to balance fine-grained permissions with usability appears to be an impossible balancing act. Android needs to overhaul the permission scheme to make it easier to understand, and app developers should plainly explain what the apps can access and how this information can be misused. Future work should determine which permission warnings and notifications are effective to users' understanding and consider alternate methods of presenting permissions to users.

## References:

Amrutkar, C., Traynor, P. and van Oorschot, P., 2015. An Empirical Evaluation of Security Indicators in Mobile Web Browsers. *IEEE Transactions on Mobile Computing*, 14(5), pp.889-903.

Baloch, R. (2016). *Bypassing Browser Security Policies For Fun And Profit*. BlackHat. Retrieved 2022, from https://www.blackhat.com/docs/asia-16/materials/asia-16-Baloch-Bypassing-Browser-Security-Policies-For-Fun-And-Profit-wp.pdf

Baloch, R. (2020, October). *Multiple Address Bar Spoofing Vulnerabilities In Mobile Browsers*. miscellaneous ramblings of an ethical hacker. Retrieved July, 2022, from https://www.rafaybaloch.com/2020/10/multiple-address-bar-spoofing-vulnerabilities.html

Boksasp, T., & Utnes, E. (2012, June 12). *Android apps and permissions: Security and privacy risks*. Norwegian University of Science and Technology. Retrieved July 30, 2022, from https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/262677/566356_FULLTEXT01.pdf?sequence=1

*Chaitrali Amrutkar, Kapil Singh, Arunabh Verma, Patrick Traynor. VulnerableMe: Measuring Systemic Weaknesses in Mobile Browser Security*

Cimpanu, C. (2016, October 28). *Bug Bounty Hunter Launches Accidental DDoS Attack on 911 Systems via iOS Bug*. Softpedia News. Retrieved July 2022, from https://news.softpedia.com/news/bug-bounty-hunter-launches-accidental-ddos-on-911-systems-via-ios-bug-509738.shtml

Duo. (n.d.). *Can Duo's Remembered Devices feature work if third-party cookies are blocked?* Duo Support. Retrieved 2022, from https://help.duo.com/s/article/2189?language=en_US

Google. (n.d.). *(Safe) Safe Browsing Testing Links*. Test Safe Browsing. Retrieved July 2022, from https://testsafebrowsing.appspot.com/

Hluchy, L. and Habala, O., 2016. Enhancing mobile device security with process mining. 2016 IEEE 14th International Symposium on Intelligent Systems and Informatics (SISY),.

Jog, Y., Damle, P., Tambulwadkar, A., & Konar, M. (2017). *Understanding Mobile Apps and Related Permissions for Android Platform*. NADIA. Retrieved July 30, 2022, from https://article.nadiapub.com/IJAST/vol105/4.pdf

Kondracki, B., Aliyeva, A., Egele, M., Polakis, J. and Nikiforakis, N., 2020. Meddling Middlemen: Empirical Analysis of the Risks of Data-Saving Mobile Browsers. *2020 IEEE Symposium on Security and Privacy (SP)*,.

Vashisht, S., Gupta, S., Singh, D. and Mudgal, A., 2016. Emerging threats in mobile communication system. 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH),.

Hardcastle. (2022). *Predator spyware sold with Chrome, Android zero-day exploits to monitor targets*. The Register. https://www.theregister.com/2022/05/24/predator_spyware_zero_days/

https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/