CordovaConfig: A Tool for Mobile Hybrid Apps' Configuration

Abeer AlJarrah
San Francisco State University
San Francisco,CA
abeer@sfsu.edu

Mohamed Shehab
University of North Carolina at Charlotte
Charlotte, NC
mshehab@uncc.edu

ABSTRACT

Despite their recentness, hybrid mobile apps have established an increasing share in the mobile apps market. This can be attributed to the fact that these apps offer the balance between providing full functionality at an affordable development cost. Hybrid mobile apps are web apps hosted in a thin native container. Several libraries facilitate building hybrid apps by providing interfaces through which native phone resources can be accessed using Javascript code. Configuring mobile hybrid apps properly is an important but often neglected activity. Coarse-grained configurations and risky default settings result in several privacy and security breaches. Moreover, middleware libraries provide a basic interface to the developers which may drive them off from changing the default settings. We are seeking to provide an automated, interactive, and contextual support for configuring hybrid apps. In this paper, we present a tool prototype, CORDOVACONFIG, which provides fine-grained configurations that are aligned with the app's behavior. We evaluate the potential use and effectiveness of CORDOVACONFIG on 22 students. Our results demonstrate that interactive configuration support can (1) help address this important non-functional requirement early in the development cycle (2) increase programmers awareness in potential risks associated with insecure settings (3) increase developers understanding of configuration items. This is supported by a quantitative and qualitative evaluation. We also uncover common programmers practices and perceptions of hybrid apps security & configurations

CCS Concepts

•**Security and privacy** → **Web application security;** *Software security engineering;*

INTRODUCTION

The number of developers involved in the development of mobile apps reached 12 million in 2016, according to Evans Data. This number is more than half of the total worldwide population of 21 million developers. It is expected to exceed 14 million by 2020 [7]. With this massive shift on demand of labor force toward mobile apps development, the need to support developers with tools to help build secure apps is essential.

In their latest report of mobile top 10 risks, OWASP identified "Improper Platform Usage" as number one risk in mobile apps development. This includes misuse of a platform feature or failure to use platform security controls resulting in an easy exploitability and a severe impact [15].

Smartphone apps are not as trusted as web nor desktop applications. Research [5] shows that users are more concerned about privacy on their smartphones than their laptops and they are more apprehensive about performing privacy-sensitive and financial tasks on their smartphones than their laptops. Improving smartphone apps security and privacy cannot be achieved without involving the app developer in the process.

The lack of application security skills, tools ,and methods are ranked as being one of the top three challenges faced by developers to maintain apps' security [10]. The lack of skills, tools and methods induces the need to provide training and increase awareness among developers to help implement better development practices.

Coding derives a cognitive burden on developers, mobile apps' development is no exception. Productivity bottlenecks divert efforts to repairing errors rather than learning to avoid them. Developers are constantly under severe time pressure; therefore, they seek the fastest ways to get the job done.

Integrated Development Environments (IDEs) and tools support have been playing a vital role in helping developers avoid errors and improve productivity. Nonetheless, certain types of logical errors resulting from poor coding practices such as misconfiguration and not following platform recommendations, are as harmful as other syntax and semantic errors. This category of errors may cause security breaches. Previous research [23] mentioned that there is a disconnect between developers' conceptual understanding of security and their attitudes regarding their personal responsibility and practices for software security. Many developers regard configurations as non-functional and not as important as the core function of the code, not to mention that the impact of such flaws may not necessarily interfere with the program logic but only surface in the course of security breaches.

In 2008, PhoneGap or Cordova, the leading mobile hybrid platform, was introduced to the community as a solution to several challenges facing the mobile development industry such as market fragmentation and the high cost of development and training. Hybrid mobile platforms can target several

MUM '18, November 25–28, 2018, Cairo, Egypt

DOI: https://doi.org/10.1145/3282894.3282931

operating systems using the same code base. The platform is supported by libraries that implement the bridge that enables access device-specific features using Javascript. Cordova Library is a middleware that is a common component in many popular hybrid platforms such as PhoneGap, IBM Worklight, App Builder, Sencha, Monaca, and Appery.io. This component is the real enabler of connecting the two worlds (Web and Native) inside a hybrid app. Recent statistics in Google Play show that hybrid apps constitute a 5.84% of the market share. Some apps are able to attract a customer base of 10,000,000+ [2]. Business, Medical, and Finance apps are at the top of the list of hybrid apps.

Adoption of the technology have faced some challenges due to the lack of tooling support [8]. Later, several platforms have invested not only on providing tooling support but also in incorporating other technologies, such as cloud-based builds. Most of the support is toward hard-core coding, debugging, and deployment. App configuration support is not only a low priority but also suffers limitations such as risky default settings and basic XML based interface. The current platform adopts command line interface to interact with the app. The app configuration can be changed by editing a text file (`config.xml`) that contains XML formatted settings.

In this work, we are proposing CORDOVACONFIG. A web-based tool to help developers configure their apps securely based on the app behavior. The tool aims to minimize the burden on the developer by providing generated configurations based on what has been monitored. The tool does not only generate configuration settings but also plays an educational role by explaining each configuration setting meaning and consequences. Educating developers on secure programming techniques through IDE proved to be effective means of raising awareness among developers [21][18][22][24]. Generating configurations values based on observed app behavior is essential to remove some burden from developers shoulders and help control app behavior at the same time.

Contributions:

- We present behavior-based configuration for hybrid apps.
- We design and build CORDOVACONFIG, a tool for configuring mobile hybrid apps.
- We design, implement , and conduct a user study to test the tool efficacy and usability
- We explore developers perceptions and common practices related to configurations of mobile hybrid apps.

This paper is organized as follows: we summarize related work that has been done in mobile hybrid apps security and previous research in incorporating security in IDEs and platforms to help users produce secure products. Then, we demonstrate the basic architecture of behavior based configuration for hybrid apps. After that we present the tool CORDOVACONFIG to offer an interactive environment for developers to configure hybrid apps. After that, we demonstrate the user study flow and design. The results of the study are explained followed by the final conclusion.

RELATED WORK

Supporting developers to maintain privacy-protective behavior when coding has been addressed in literature in different aspects. Rebbeca et al.[3] highlight key challenges developers face in following best practices related to privacy. Being a secondary concern, privacy is not a priority to developers. Not to mention the difficulty of reading and writing privacy-related policies. In fact, according to Zibran et al. [25], API documentation is among the top factors that negatively affect API usability. Mobile hybrid platforms are no exception. Thus, solutions to solve these problems have been directed towards following a human-centric approach in API design not only to improve usability but also to promote security and privacy earlier in the design process. Myers et al. [14] provide insights into understanding how different API stakeholders (API designers , API users, and API consumers) have different needs - contradictory at some point - which affects the way the API is designed and built. They stress the importance of adopting a human-centric approach to promote usability. Having the API user in consideration when designing the API and investing in tooling support may positively affect API usability. Green and Smith [9] have also supported the argument of creating developer-friendly and developer-centric approaches to promote usable security. They have identified ten principles for creating usable and secure APIs. Examples include making the API easy to use, even without documentation and ensure that defaults are safe and never ambiguous.

Along the same lines, implementing secure code has been receiving a demonstrable attention recently. Practices followed by developers, such as lapses of attention to security and the lack of application security knowledge in general, are main reasons motivating the direction toward providing more support to developers in order to produce secure code. Jing et al. [21, 20] propose a tool named ASIDE (Application Security IDE), an Eclipse plugin for Java. The plugin goal is to help programmers prevent errors by providing interactive code annotation and refactoring. The tool evaluation [22] reveals that programmers are only likely to successfully address such non-functional errors during programming if the effort and the cognitive burden is sufficiently small. Moreover that programmers need easy ways of understanding and learning about secure programming regardless of their experience level.

The need to provide a detailed context-rich interface for system configurations has been already discussed for different technologies. Raja et al. [17] have discussed the effect of Windows Vista personal firewall basic interface on users mental model. The UI does not present enough details which may result in users developing an incorrect mental model of the protection provided by the firewall. They suggest a prototype to support the development of a more contextually complete mental model through the inclusion of network location and connection information. The study shows that participants produce richer mental models after using the prototype. In their analysis of the causes of software errors in terms of chains of cognitive breakdowns, Ko and Myers [13] indicated that certain type of errors is due to attentional issues such as forgetting or a lack of vigilance.

BEHAVIOR-BASED CONFIGURATIONS & HYBRID APPS SECURITY

Adobe's PhoneGap is by far the fastest growing platform and the most widely used to develop hybrid apps [16]. This platform uses Cordova library to enable the communication between the web side and native side. The current distribution of the library suffers several security limitations related to its configuration scheme. Mainly, having coarse-grained settings and risky default values.

Code injection attacks are the most prominent threat to hybrid apps [11][12]. The malicious injected code can abuse the bridge provided by the library to access device native resources (plugins) such as camera, geolocation, and contacts. Malicous code may also target the app itself as it can maliciously tamper the app logic by redirecting to an unwanted page. Table 1 summarizes how improper configurations can maximize damage resulting from code injections attacks. An analysis of

Table 1: Configurations Issues and impact on apps' security

Configurations Issues	Impact
Not having a proper CSP	Enable malicious code to be triggered/ executed
Global plugins declaration	- Plugins can be accessed in all app pages/states - Having unnecessary plugins (over-privileged)
Risky defaults	- Resource network access is set to "*" - Allow calling built-in apps such as SMS, TEL, Maps

these issues and a proposition of automated behavior-based configurations for hybrid apps were addressed in a previous research [1]. Automated behavior-based configuration aims to secure apps by generating proper configurations based on monitored app behavior. It also aims to help developers as it automates the generation of an initial set of configurations to start with.
The approach adopts a more detailed granular level of configurations. It defines the app by its states rather than the whole app, hence, aligns resources accesses to finer-granular level. The approach also adopts least privilege principle in granting access to the device sensors or URLs. Aligned configurations to app behavior are not the only benefit of this approach, minimum effort and time burden on developer's shoulder is also needed given that most developers spend minimum time on configurations and security aspects when developing.

The tool in this work feeds on the generated set of configurations using the behavior based configuration. Moreover, it heavily involves the developer is the process of composing the final set of configurations. This is needed not only to involve developers and increase awareness but also to reduce false positives/negatives accesses resulting from the dynamic approach.

CORDOVACONFIG: A TOOL FOR HYBRID APPS' CONFIGURATION

We have developed CORDOVACONFIG, an interactive web-based tool that allows interaction with developers to configure hybrid apps. The tool is built on the top of the instrumented Cordova library mentioned in Section 3. CORDOVACONFIG walks the developer in a wizard style where there are different stages of app configuration to finally generate what the developer determines to be a valid behavior. The tool also presents an educational part in the beginning where developers get explanations of the meaning of the configurations and the impact of keeping the current values. The developer can download the configuration file and use it instead of the default one. Figure 1 explains the tool work flow. The tool directs the developer through several steps before generating configurations.

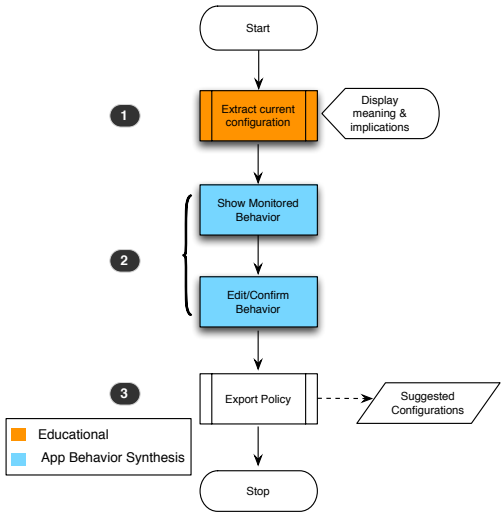

Figure 1: CORDOVACONFIG Work Flow

Educational Phase

Educating users is a pillar in usable security. Thus, we have designed the tool to include an educational part to help improve developers' understanding of configuration items. The tool starts with reading the current configurations in `config.xml` file. Then, it generates an explanation in terms of the meaning of each configuration item and the impact of having these configuration values in the app. For instance, Figure 2 demonstrates a set of configurations (including resource network access and navigation whitelist) configuration found in the `config` file and the corresponding meaning/impact of such value. It also captures the android permissions found in `AndroidManifest` file and explains the consequences of using such permissions. There exist several parts (pages) of this educational phase to go over all the configurations values.
The goal of this part is to raise awareness in Cordova based apps' configurations meaning, and their impact.

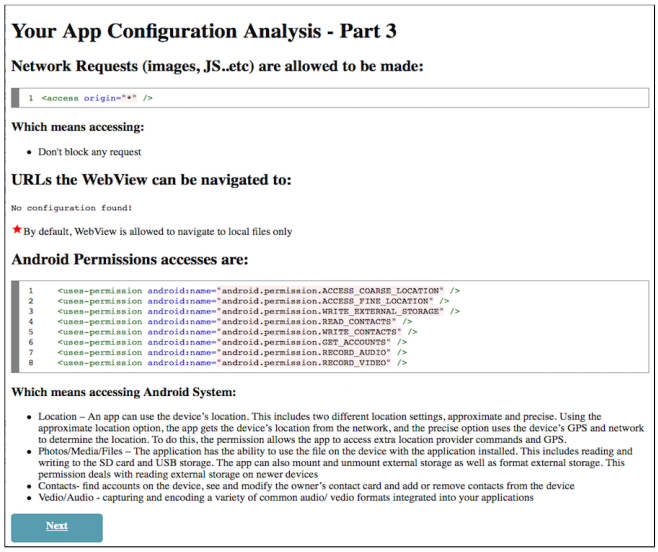

Figure 2: Explaining scanned configurations meaning/impact

App Behavior Synthesis Phase

In this phase, the tool shows the apps states along with the monitored resources accesses. Follows a detailed explanation of what defines an app behavior.

App States

As explained in the previous section, the behavior based model captures app states. Each app state is associated with a set of plugin access(es). Each state is also captured with a screen shot

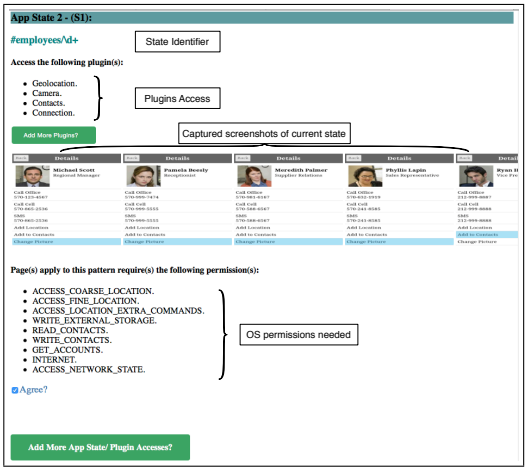

Figure 3: Plugin accesses and OS permissions captured *per* state

of the app when using the plugin. A list of plugins accessed, URLs accessed, and the corresponding Android permissions needed to be granted for these plugins are all captured per state, see Figure 3 . The developer has the option to: add any plugin that has not been captured or delete any unnecessary access to a plugin. The tool also enables the developer to define a new state. Since the tool"s feed of information about the monitored app behavior is based on dynamic analysis, false negatives are presumed. Involving the developer is necessary not only to eliminate false negatives but also to give the developer more control to tweak the configurations in a simple way.

App States Transitions

The tool captures the app state transitions and translates them into configurations to control app flow against attacks trying to tamper the app control flow. The captured states transition is displayed to the developer in the form of a state diagram (see Figure 4. Approving it creates configurations in a State Chart XML (SCXML) format that represents the app flow. The developer can change the configurations generated to any

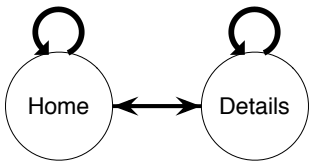

Figure 4: State Transition Example

other form she chooses to. Identifying app states and their transition may help the developer better understand the app behavior and the new configuration scheme suggested.

App interactions with external entities

Another important aspect of modeling the app behavior is to capture the app interactions with external entities, such as:

- Other Applications (`Intents` in Android).
- Remote network resource access
- Navigation URLs within the app

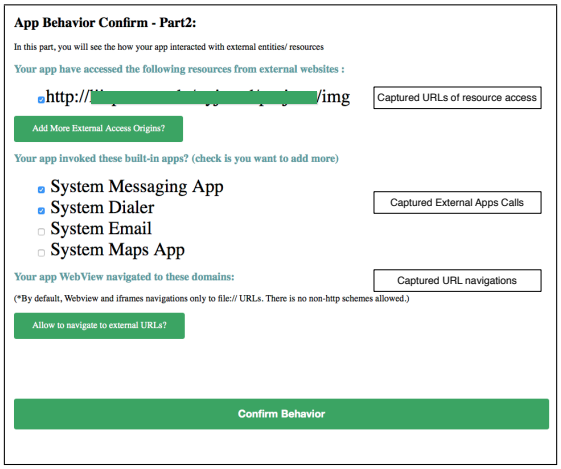

Figure 5: App interaction with external entities

Figure 5 shows a sample of the settings related to interactions with external entities. The developer can also add/modify/delete any of these settings then confirm them.

Generating Configurations & Permissions Phase

After the developer confirms the behavior, configurations are generated to align with it. As shown in Figure 6, generated settings are comprised of three parts:

- Configurations to control plugins, URLs, Navigations and interactions with other apps per state.
- Configuration for app state transition, which is a new configuration item added by our tool to control app transitions.
- Android permissions per state, which can be added to the `Android-Manifest.xml` file to control the native side of the app.

Our prototype implementation supports interactive environment for developers to help further customize the configurations in case the observed behavior did not capture all required features.

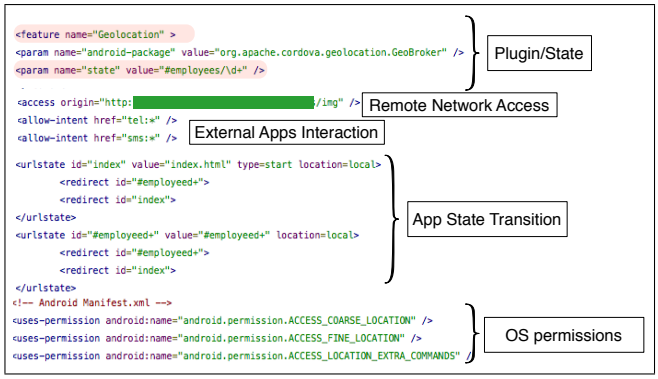

Figure 6: Generated Configurations

Case Study App

The app presented in this study is a simple Employee Directory application. It is referred to as `empDir`, which views information about a list of employees through a summarized list item. Clicking on a list item displays a detailed information about that employee. It also enables the user to perform the following functionalities:

- Getting the current location of the device. This functionality requires access to the Geolocation API.
- Changing the employee profile photo. This functionality requires access to the Camera API.
- Saving the employee contact information into the device contacts list. This functionality requires access to the Contacts API.

The app (see Figure 7) is taken from one of the PhoneGap tutorials[6]. This app is chosen for the following reasons :

- Utilizes several APIs such as Geolocation, Contacts, and Camera.
- Interacts with other Apps (Dialler & Messages)
- Loads resources from an external URL.

This single page app makes a good candidate for an app that needs to customize configurations for almost all configurations items found on `config.xml`. It also utilizes the state identification and aligned configuration per state. The app starts with

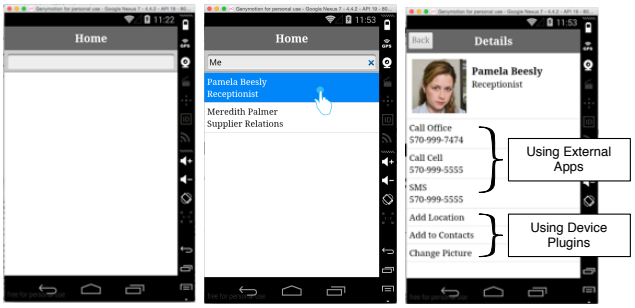

Figure 7: Employee Directory App

a home page that has a search text. The user can enter the name of the employee which will filter the list of the employees to match the text. Once the user selects an employee, a detailed information page opens. The page enables contacting the user through calls and SMS. It also enables adding a location for the user, adding the employee phone numbers to the device contacts and changing the current picture of the employee using the device camera.

USER STUDY ON HYBRID APPS DEVELOPERS

The current CORDOVACONFIG prototype is a proof-of-concept web-based tool for generating configurations based on the observed the behavior of an app. The goal of this tool is to allow the interaction between the developer and the tool to address an important and often overlooked step, that is app configurations.

The main goal of the user study is to explore developers perception of the tool and its value. A controlled repeated measure study is conducted to either prove or refute the following hypotheses:

- H1: Using CORDOVACONFIG changes developers' mental model in regard to understanding the boundaries of the required privileges of an app.
- H2: Using CORDOVACONFIG improves developers' understanding of a specific app' configurations.

The study also further explore the following questions:

- Q1: What is the developers' perception of risky settings
- Q2: What is the developers' perception of the value of CORDOVACONFIG

In order to test the hypotheses, we identify the variables and the research instruments used to measure them (see Table 2). Questions used on the Pre/Post Survey and during the onterview can be found online [1]. The user study is a controlled repeated measure experiment. Metrics for H1 and H2 are set to be the score of the correct answers in the Pre/Post Survey. Tool usability is also measured using SUS index [19]. The last two questions aim to explore unknown aspects, so we use qualitative data points to understand the current situation of developers' perceptions.

[1] https://github.com/aaljarra/CordovaConfig/

Table 2: Variables Measurement Methodology

Variable	Research Instrument	
Understanding of an app configurations	Pre/Post Survey	Questions - 1,2,3,4,8
Changing developer's mental mode	Pre/Post Survey	Questions - 5,6,7
Tool usability	System Usability Scale (SUS) Survey	
Perception of risky settings	Interview	Question 3
Perception of the tool value	Interview	Question 2

Recruitment

We have recruited 22 students who have a background in mobile application development and Cordova/PhoneGap development. We have used flyers and emails to call for participation. Participants are rewarded $20 gift cards in return of their time and effort. Most of the participants are either current or previous students of a Mobile Application Development course focusing on Android. In addition, the course also offers material focusing on hybrid apps development by introducing students to the PhoneGap/Cordova library. The assignments given to the student related to this subject focuses on familiarizing students with using Cordova built-in plugins. In addition to other web-based platforms specific to mobile apps.

User Study Protocol

The participants are provided with the app `empDir` that is mostly implemented missing only the third functionality - adding to contacts. Participants are asked to resolve what is missing through testing the app and checking the app files. The app configurations are kept to default settings. Once the participant is able to detect the missing functionality. She is asked to implement it. On average, each participant took 45 minutes to finish this step. To be able to finish the task, participants need not only to write the code for the missing functionality -calling plugin API on JS side- but also to configure the app properly so that it includes the declaration of the contacts plugin API. The goal is to familiarize the participant with the app code and its logic. The participants are asked to test the app and check if it satisfies all requirements. At this point, participants experienced a typical hybrid app development cycle. Then participants are asked to answer a pre-survey. After that, participants are provided with the same app built on the instrumented version of the library. Thus, all app transactions are monitored. Participants are asked to test this app to make sure it's working properly, meanwhile, the tool can have more data on the app behavior by logging all app interactions and transitions. Then, participants are asked to use CORDOVACONFIG. After that, participants are asked to answer a post-survey. Finally, participants are interviewed. The steps of the procedure can be summarized as follows:

1. Complete the work on `empDir`.
2. Answer Pre-Survey
3. Test `empDir` on instrumented cordova library
4. Use CORDOVACONFIG
5. Answer Post-Survey
6. Interview

To cancel any bias, two versions of Pre and Post surveys (Pre/Post Survey A & Pre/Post Survey B) are formulated such that questions in Pre-Survey A are used on Post-Survey B and vice versa. Participants are randomly assigned to answer either Pre/Post Survey A or Pre/Post Survey B. Both surveys aim to measure the same parameters but through different questions. The interview at the end of the study is needed to measure qualitative variables such as developer perception of the tool and their awareness in hybrid apps' security. Each interview lasted 5 to 15 minutes.

USING CORDOVACONFIG

We have created a shortcut for the tool URL on the computer desktop. Each participant is asked to click on the icon to start the tool. The average time participants spent using the tool is $\approx$ 15 minutes. Only 2 participants asked questions while using the tool. The questions they asked were about how to use the tool.

The final output of the tool is the generated configurations. All participants indicated that they would use the generated configurations and the Android permissions rather than keeping the old ones, mainly because they indicated that the configurations are precise and exact and conform to the app's functionality.

To observe how much attention participants would pay to configurations when using the conventional method, we have changed the configurations of the app. We have added unnecessary plugins but also removed the needed plugins for the app to function. By the end of the development testing, we have noted that all participants added the required plugins, but NO ONE has removed the unnecessary plugins. This was a strong signal that developers tend to change configurations by adding the required settings, but rarely they do any checking or revision of the setting values as long as the app is working.

RESULTS

Below a detailed demonstration of the participants' background, data analysis procedures, and interpretation of research results.

Participants Demographics

The education level of participants varied between graduate 18 (81.8%) and undergraduate 4 (18.2%) aging between 21 - 32 years old. Most of the participants are male comprising 90.9% (20) as opposed to 9.1% (2) female participants. Table 3 shows a summary of the participants' development experience in three areas; Mobile Applications, Web Applications, and Mobile Hybrid Applications.

As this tool is designed to maintain app's security through aligned configurations, we have asked the participants to self

Table 3: Participants Development Experience

Experience\# Years	<1	1-3	3-6
Mobile Based	68.2% (15)	22.7% (5)	9.1% (2)
Web Based	31.6% (7)	54.5% (12)	13.6% (3)
Mobile Hybrid Based	95.5% (21)	4.8% (1)	-

assess their level of security knowledge (Novice, Intermediate, Expert). Most of the participants 86.4% (19) have indicated a Novice level while 13.6%(3) have indicated an Intermediate level.

Overlooking security and dealing with it a secondary issue is

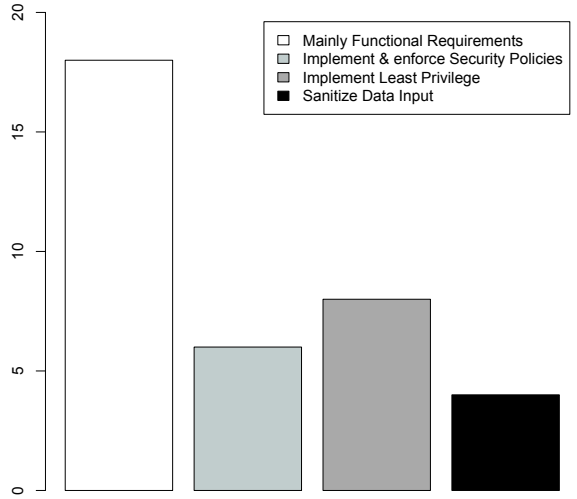

Figure 8: Common Coding Practices followed by participants

a common practice among developers. To check if this also applies to this group, we asked them to check what applies as a common coding practice. Figure 8 depicts the options and shows that, as expected, they normally focus on the functional requirements of the app more than any other security-related practice.

H1: CordovaConfig **& Configurations Understanding**

Participants answered a set of questions before and after using the tool. The questions focus on measuring developers understanding of configuring the app to satisfy specific requirements. Specifically, settings related to :

- Plugin usage
- Network access
- Platform permissions
- Interaction with external apps

Participants are also asked to provide an explanation of certain settings impact, and how-to change configurations to satisfy a specific need. The answers to these questions are collected. The average of scores before using the tool is 3.4 while it changes to 4.4 after using the tool. Figure 9 demonstrates the participants' score in the Pre/Post survey questions related to this variable. We can see that most participants are able to score higher in the post-survey. Nonetheless, there are 4 participants who had a negative effect.

A dependent t-test (within-subjects) is conducted using par-

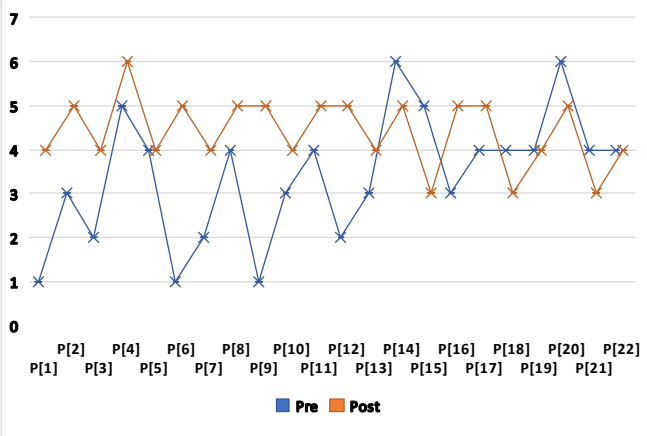

Figure 9: Developers' Configuration Understanding Scores

ticipants' scores before (*M=3.4, SD=1.47*) and after (*M=4.4, SD=0.79*) the survey. The result supports the hypothesis that developers are more likely to understand the purpose and the meaning of the app configuration items after using this tool, $T(21)=7.96$, $p= 0.01$, $p < .05$.

H2: CordovaConfig **and Developers Mental Model**

One of the features provided by most hybrid platforms together with Cordova, is to allow inter-applications interactions between apps. To perform an operation that requires calling another app, the developer needs to use URIs. For instance, to dial a number developer can use this URI:
`<a href="tel:1(xxx)xxx-xxxx">Call Me!</a>`.
Clicking this link launches the native Dialer app and passes the number. Other native apps such as maps, SMS, and mail can be called the same way. Default configurations allow interaction with all these apps.

Favoring simplicity at the expense of security might be the reason why default configurations are kept this way. However, it is important to recognize that this violates the principle of least privilege. Moreover, it may invite a misconstruction at the developer end in terms of understanding the boundaries of the hybrid app. Developers need to recognize that having maps functionality in the app - for example - may require calling the built-in maps app but this should not require the app to have any permissions related to maps functionality. Developers needs to recognize that once the maps app is open the control is transferred to the maps app which is outside the boundary of the app. Developer need to understand that they do not need to include any platform permission related to geolocation as this is outside the boundary of the app. Thus, in the survey, we ask questions to test the tool effect on changing the developer mental model by asking what permissions, configurations, and

changes needed to call native apps. The questions addressed the case when the app calls another app and when the app itself uses a functionality and how this should differ in terms of what permissions to ask for. The average of scores before using the tool is 0.6 while it changes to 1.22 after using the tool. A dependent t test (within-subjects) using participants' scores before (*M= 0.5 ,SD=0.7*) and after (*M= 0.1 , SD=0.8*) the survey is performed. The result supports the hypothesis that developers are more likely to understand the boundary of the app and to recognize the needed permission/configurations needed, *T(22)=4.44, p= 0.047, p <.05* .When we categorized participants based on common coding practices they follow (Figure 8), we have a found two groups have a significant relationship with this variable. Namely, participants who checked that they usually implement and enforce security policy (*T(1,5)=25, p= 0.004, p <.05*) and participants who checked that they would sanitize external input (*T(1,3)=27, p= 0.014, p <.05*).

Q1: Developers' awareness of potential risks

The goal of using the tool is to generate a set of aligned and fine-granular configurations rather than using the default configurations. To examine the difference the tool is making developers are asked to compare the configurations generated by the tool and the default configurations of `empDir`. Differences include :

- The inclusion of only the plugins that are necessary.
- Setting specific values of allowed URLs for resources' loading rather than the default value '*' that allows all domains
- Setting specific values for types of apps allowed to be launched from the app

When asked, most participants have noticed the first two. Then when they were asked to indicate the implications of keeping the default configurations in the app rather than using the generated configurations, they gave different answers which shows the different perspectives of understanding. Figure 10 and 11 show the distribution of developers answers. Although most developers are able to indicate a security related risk of keeping the default configuration, only *four* participants were able to mention attack types that can potentially compromise an app. The rest were unable to describe nor mention a scenario that can put an app under risk.
Previous work [4] postulates three reasons for "Why Good People Write Bad Code":

- Technical factors which mean the underlying complexity of the task itself.
- Psychological factors including poor mental models or difficulty with risk assessment.
- Real-world factors comprising lack of financial incentives and production pressures

While our results similarly highlight a number of real world constraints, we identify that a key inhibitor toward secure software development practices is an " it's not my responsibility" attitude.
Hybrid platforms have improved over time. Current distributions have a content security policy (CSP) to restrict local

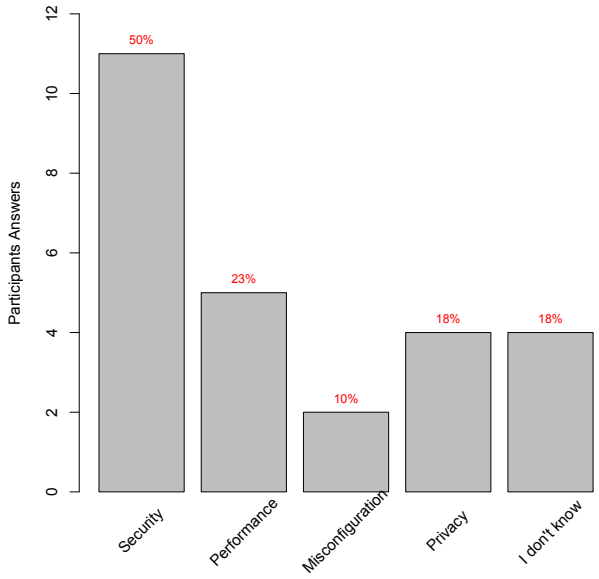

Figure 10: Perceived Implications of having unaligned plugins settings

access per page. Developers are required to understand CSP meaning and learn how to customize it when needed. We have asked the developers if they know what is CSP or what it stands for. Only *one* participant was able to recognize it's meaning.
Then, we asked if they would normally check the configuration of the app. Only *one* developer indicated that she would check the configuration file. All other developers indicated that they don't normally check the file for many reasons including lack of time, lack of understanding, security is not a priority, and for many of them, it is not necessary as long as the app is working properly.

Q2: Perception of the benefits of CordovaConfig

All participants indicated that they are likely to use the tool to configure their apps. One indicated that this tool is most beneficial when prototyping apps. By asking about the benefit/value of this tool, we show answers distribution in Figure 12. Easiness is the most observed benefit of the tool, depicted by answers such as:

> [P7] "It is very tough to code hybrid apps, there is plenty of stuff to search and look for. Having this would make it less hectic".
> [P3] "Managing permissions is easy now".

Many participants indicate that using this tool would enhance the security of their apps as well, example answers include:

> [P1] "It restrict access to URLs, installed apps and plugins".
> [P4] "It automates the configuration according to

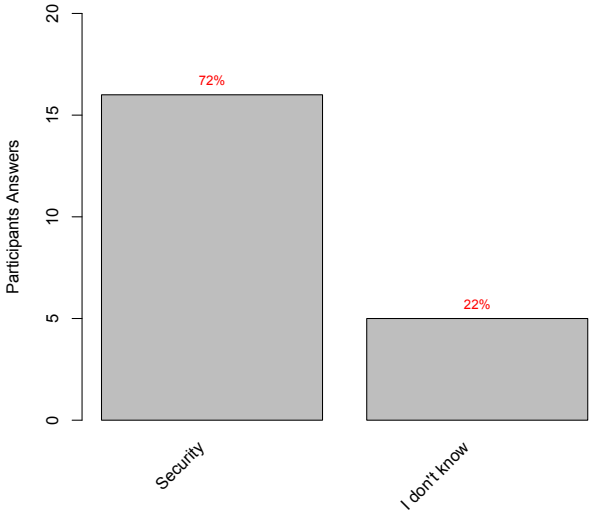

Figure 11: Perceived Implications of having unaligned Network Access settings

app behavior so that your app is secure".

Many participants indicate that this tool has increased their understanding of hybrid platform configurations in general and the app configurations in specific. Examples of participants responses include:

> [P13] "It helped understand the app flow".
> [P6] "It enhances readability of Cordova configurations".
> [P16] "I was not aware of these settings and what they do. I am more aware now and I feel I have more control".
> [P19] "I appreciate all descriptions and explanations since we programmers tend to shorthand things easily".

A few participants indicated that using this tool gave them more control over the app. Example comment:

> [P13] "It seems like it is a good tool to insure that the app do what is supposed to do".

Only two participants described generating page-wise level settings as a value of this tool, such as:

> [P10] "Sometimes we don't need plugins in certain pages, this generates configurations per page".

On the whole, participants showed high level of excitement, interest and gratitude while and after using the tool. This can be attributed to the change from the original configuration process or even to the idea of bringing this whole step to their attention.

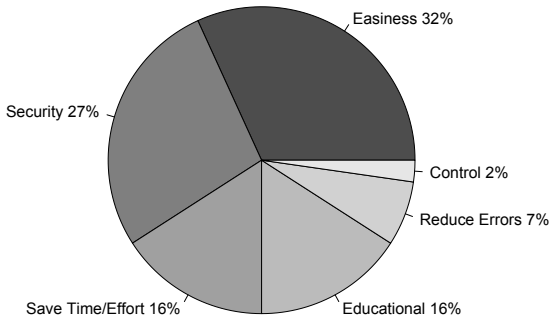

Figure 12: Perceived Benefits of CORDOVACONFIG

CORDOVACONFIG **Usability**
Measuring usability of a tool is challenging as it is a subjective matter that varies among different users. However, it is essential for assessment as users satisfaction of the tool determines its' success. We use the industry-standard System Usability Scale (SUS) [19] to evaluate the usability of our CORDOVACONFIG. Based on SUS formula, we generate an overall SUS score of 86.25 for the tool. According to prior research, a SUS score above 68 is above average for general applications.

STUDY LIMITATIONS & FUTURE WORK

While this study provides insights regarding usability and efficacy of a commonly overlooked aspect of hybrid apps development, our work has a main limitation . The small number of participants and the fact that they are mostly students may limit the findings of the study. Future research that targets a different group of developers is needed to better evaluate the tool and to explore different perspectives.

CONCLUSION

Developers are continuously under pressure to deliver code in a short amount of time. Their peccability is as expected as any other calculable risk. While it is hard to prevent developers from making mistakes, it is attainable to change the way they interact with the programming environment. This work is a proposition toward providing developers a support tool to overcome a common overlooked need.
CORDOVACONFIG is a tool that helps developers configure Mobile hybrid apps. The tool generates initial settings based on app behavior, yet developer can edit the settings if needed. Our study results showed that the tool is a usable option. The tool is also able to increase developers knowledge in secure configurations. The user study showed that none of the developers would change the default settings in addition to lacking knowledge and interest on potential risks and attacks in addition to the security policies offered. Thus, we argue the vital need for a tool to fill this void.
Considering the novelty of the platform, this user study has

a limitation. As the case with similar studies, finding participants who have the required background needed to conduct the user study was the main concern. Thus, most of the participants we were able to recruit are novice programmers in hybrid mobile apps in general and Cordova-based apps in specific. Having professional developers evaluation of this tool would provide valuable insights which we are working on as a future extension.

This work is a step towards providing more tooling support to developers of hybrid mobile apps not only to facilitate their tasks but also to help produce more secure apps.

REFERENCES

1. Abeer AlJarrah and Mohamed Shehab. The demon is in the configuration: Revisiting hybrid mobile apps configuration model. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, pages 57:1–57:10, New York, NY, USA, 2017. ACM.
2. AppBrain. Phonegap / apache cordova. http://www.appbrain.com/stats/libraries/details/phonegap/phonegap-apache-cordova.
3. Rebecca Balebako and Lorrie Cranor. Improving app privacy: Nudging app developers to protect user privacy. *IEEE Security & Privacy*, 12(4):55–58, 2014.
4. Andrew Blyth. Secure coding principles and practices. *Infosecurity Today*, 1(3):46, 2004.
5. Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. Measuring user confidence in smartphone security and privacy. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 1. ACM, 2012.
6. Christophe Coenraets. Tutorial: Developing a phonegap application. http://coenraets.org/blog/phonegap-tutorial/.
7. Evans Data Corporation. Mobile Developer Population Reaches 12M Worldwide, Expected to Top 14M by 2020. https://evansdata.com/press/viewRelease.php?pressID=244.
8. Progress Software Corporation. The state of hybrid mobile development. https://developer.telerik.com/featured/the-state-of-hybrid-mobile-development/.
9. Matthew Green and Matthew Smith. Developers are not the enemy!: The need for usable security apis. *IEEE Security & Privacy*, 14(5):40–46, 2016.
10. SANS Institute. 2016 state of application security: Skills, configurations and components. https://www.sans.org/reading-room/whitepapers/analyst/2016-state-application-security-skills-configurations-components-36917.
11. Xing Jin, Xuchao Hu, Kailiang Ying, Wenliang Du, Heng Yin, and Gautam Nagesh Peri. Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*.
12. Xing Jin, Tongbo Luo, Derek G. Tsui, and Wenliang Du. Code injection attacks on html5-based mobile apps.
13. Andrew J Ko and Brad A Myers. A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages & Computing*, 16(1):41–84, 2005.
14. Brad A. Myers and Jeffrey Stylos. Improving api usability. *Commun. ACM*, 59(6):62–69, May 2016.
15. OWASP. Mobile top 10 2016-top 10. https://www.owasp.org/index.php/Mobile_Top_10_2016\protect\discretionary{\char\hyphenchar\font}{}{}Top_10.
16. PixelCrayons. Cross-platform mobile development: Trends, tactics and tools. https://www.pixelcrayons.com/blog/mobile/cross-platform-mobile-development-trends-tactics-and-tools/.
17. Fahimeh Raja, Kirstie Hawkey, and Konstantin Beznosov. Revealing hidden context: improving mental models of personal firewall users. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 1. ACM, 2009.
18. Joseph R Ruthruff, Shrinu Prabhakararao, James Reichwein, Curtis Cook, Eugene Creswick, and Margaret Burnett. Interactive, visual fault localization support for end-user programmers. *Journal of Visual Languages & Computing*, 16(1):3–40, 2005.
19. Usability.gov. System Usability Scale (SUS). http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.
20. Jing Xie, Bill Chu, and Heather Richter Lipford. Idea: interactive support for secure software development. In *International Symposium on Engineering Secure Software and Systems*, pages 248–255. Springer, 2011.
21. Jing Xie, Bill Chu, Heather Richter Lipford, and John T Melton. Aside: Ide support for web application security. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 267–276. ACM, 2011.
22. Jing Xie, Heather Lipford, and Bei-Tseng Chu. Evaluating interactive support for secure programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2707–2716. ACM, 2012.
23. Jing Xie, Heather Richter Lipford, and Bill Chu. Why do programmers make security errors? In *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*, pages 161–164. IEEE, 2011.
24. Jun Zhu, Heather Richter Lipford, and Bill Chu. Interactive support for secure programming education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 687–692. ACM, 2013.
25. Minhaz F Zibran, Farjana Z Eishita, and Chanchal K Roy. Useful, but usable? factors affecting the usability of apis. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 151–155. IEEE, 2011.