Accuracy, Security, and Architecture Impacts and Challenges of Mobile and Web Technologies:Geolocation Field Data Collection in Washington State Water Resources

Jeremiah D. Miller, Sam Chung, Teresa Escrig, Yan Bai,
Institute of Technology
University of Washington,
Tacoma, WA 98402, USA

Barbara Endicott-Popovsky
Center for Information Assurance and Cybersecurity
University of Washington,
Seattle, WA 98195, USA

Jan Whittington
Department of Urban Design and Planning
University of Washington,
Seattle, WA 98195, USA

ABSTRACT
The purpose of this paper is to examine accuracy, security, and architecture impacts and challenges of mobile and web technologies through the case study of collecting geolocation field data in Washington (WA) State water resources. Effective management of water as a public resource relies on the capture, storage, and retrieval of accurate geographic position data. This is also true of a broad range of business domains beyond water resources, such as earth sciences, city planning, and navigation to name a few. Application developers must choose how to capture this information (with enough accuracy to be useful for its intended application) and get that data to a place where it can be processed and used. A traditional monotonic application using a Global Positioning System (GPS) and a mobile app using a smartphone are available today. The advent of HTML 5 now allows the development of a mobile web app, which is not dependent on any particular smartphone platform. These multiple technical options lead to two research questions: How does an HTML5 mobile web app solution work in terms of accuracy, security, and architecture, compared with a GPS-based solution and a mobile native app solution for collecting geolocation field data? And second, as HTML5 mobile web apps are a relatively new technology, what best practices can we uncover to assist in the process of choosing between an HTML5 mobile web app and a mobile native app, and also what are the best practices for building a mobile web app that can operate offline? To answer these questions, we build an HTML5 mobile web app called 'LocationSharpener' for collecting GPS locations leveraging three of the new HTML5 features: IndexedDB, AppCache, and GeoLocation. We use the mobile web app to collect locations of water resources facilities like wells and diversions and analyze how accurately the app collects the geolocation data. We apply threat risk modeling to the mobile web app to analyze its security and privacy compared to that of the native mobile app approach. In addition, by documenting the architecture of the mobile web app with multiple views, we analyze how HTML5 affects the architecture and present best practices for building a mobile web app that can operate online and offline. The analysis of location accuracy shows the HTML5 mobile web app approach provides acceptable location accuracy even when offline. Threat modeling demonstrates that, in contrast to native mobile apps, mobile web apps offer a clear advantage to users and developers: an isolated execution environment makes it more difficult for a malicious mobile web app to read private data; As an outcome of the architecture documentation we also propose new best practices of developing a mobile web app using HTML5: the developer must consider two subsystems for online and offline use cases and a sequence of connected, disconnected, and connected deployment modes. Also, standards-based web apps are portable across virtually any operating system. This also eases the burden on developers, since they can write mobile web app once and deploy anywhere.

General Terms
Mobile Computing, Security

Keywords
Mobile Web App; Native Mobile App; HTML 5; Geolocation Field Data Collection; Water Resources

1. INTRODUCTION
Each field visit to a water well or river diversion constitutes an opportunity to update the location data and other attributes for that station. Physically finding the facilities, determining the owner, and modeling the hydrologic and environmental impact of the water extraction all depend on accurate location data. Water resource agency staff and members of the public wanting to collect better locations for these facilities typically use a handheld Global Positioning System (GPS) receiver, or increasingly, a mobile phone to record the geolocation.

In order to solve the weaknesses of GPS devices for geolocation data collection, native mobile apps (e.g. using ArcGIS Mobile) have been developed for directly capturing this data into a disconnected database for later synchronization with an enterprise data management system [5]. However, such native mobile apps are also problematic as they can be costly to maintain since multiple versions of mobile apps are required for heterogeneous platforms. Also, they require a specialized skillset to develop, and must be installed as platform-specific applications on the client device. These constraints, along with licensing requirements, place such solutions out of reach for not only many software development shops but also many of their customers (e.g. due to license cost.) Moreover, whenever a mobile app is downloaded and installed on a mobile platform, it can be a potential security and privacy threat [8].

The recent adoption of support for HyperText Markup Language (HTML) 5 and associated technologies such as AppCache, IndexedDB, and Geolocation in mobile web browsers now enables a relatively attractive alternative to native mobile apps to reduce barriers to collecting geolocations into enterprise data systems [17]. Neither location awareness nor offline operation was possible in HTML4 technology. In a broader sense, we can help to spur a critical mass of developers to think seriously about how they can provide useful new web apps that leverage the whole new range of possibilities introduced by modern smartphones.

We propose a platform-independent mobile web application that simply runs in an HTML5-capable web browser. In contrast to solutions like PhoneGap[1] and mobl[2], no native app needs to be compiled or installed. We leverage mobile web browser geolocation support and HTML5 offline features to simplify the offline collection and subsequent automatic upload of geolocation for points of interest that might fall outside cellular data coverage areas. This mobile web app approach brings two research questions to be answered:

1) How does an HTML5 mobile web app solution work in terms of accuracy, security, and architecture, compared with a GPS-based solution and a mobile native app solution for collecting geolocation field data?

2) What best practices for HTML5 mobile web app development can we uncover?

For this purpose, we develop an HTML5 mobile web app for collecting the GPS locations of wells called 'LocationSharpener' leveraging three of the new HTML5 features: IndexedDB, AppCache, and GeoLocation. This LocationSharpener accesses a disconnected copy of features which is initially loaded from an online data system and then stored locally on the mobile client. In this case we will specifically use water resource features in order to demonstrate the concept, although the solution's design is applicable to many other types of data. During a field visit to a well, the user brings up a particular record on the now-offline web app and taps a button to link the device's current location to that record, which is then saved to IndexedDB. Later when the device gets back online, it will upload the data through a REpresentational State Transfer (REST)-ful web service that saves the location to the enterprise database.

By using 'LocationSharpener', we describe how the HTML5 mobile web app approach affects the ways of collecting geolocation field data for WA state water resources, compared to GPS devices. We collect locations of water resources facilities like wells and water diversions andanalyze how accurately the mobile web app collects the geolocation data. We also assess which mobile-based approach is mostsuitable for future geolocation field data collection between an HTML5 mobile web app and a native mobile app. Instead of simply describing platform interoperability, we apply threat risk modeling to the mobile web app and analyze the security characteristics of the mobile web app compared to the native mobile app approach. In addition, we propose new best practices for developing an HTML5 mobile web app: we first document the architecture of the 'LocationSharpener' in Unified Modeling Language (UML) by using the 5W1H re-documentation methodology [1]. We analyze the documents in terms of online and offline use cases and then propose how we design the architectures of a mobile web app. The analyses of location accuracy and threat modeling demonstrate that the mobile web app approach provides acceptable location accuracy even when offline and poses fewer security threats to users' private data, while the architecture documentation reveals new development guidelines.

The paper is organized as follows. Section 2 introduces background information of key concepts and technologies. Section 3 covers previous work, and Section 4 describes the development of an HTML5 mobile web app: a use case scenario of a prototype, design, and implementation are presented. Sections 5, 6, and 7 describe analyses of location accuracy, threat risk modeling, and architecture. Section 8 describes conclusion and future work.

2. BACKGROUND

2.1 Geolocation Field Data Collection

The staff members of Washington (WA) State Department of Ecology collect geolocation field data for WA water resources by bringing GPS devices with them when they visit regulated sites for other reasons such as inspection or groundwater data collection [16]. Such visits are also an opportunity to collect GPS coordinates for these sites, although that can be by itself the primary purpose of the visit. While personnel increasingly collect geolocations, social obstacles the inconvenience or expense associated with using dedicated GPS devices and transferring the coordinates to a public water resource database can easily dissuade users: waypoint identifiers that can be used to distinguish location records are entered into the devices through interfaces that are not user-friendly. Back in the office, the collected data are then manually typed into a database system. In some cases, the GPS devices cannot collect the geolocation data because of obstructions like trees blocking the signals. We estimate that 10% of locations cannot be collected with a GPS device due to this issue. Yet it is important to collect some location data in order to improve the data quality in the enterprise database: most of the existing records have very low accuracy (e.g. 0.25m average error). This very low accuracy of legacy location data stored in an enterprise water resource database for small-scale water supply infrastructure like wells, water meters and river diversions, leads to both environmental and cultural costs as a result of the inhibition of effective management of limited water resources [5, 11, 15].

2.2 HTML5

HTML5 is a collection of draft and final specifications and technologies enabling the creation of browser-based applications that do not require special plugins to create a rich user experience [17]. The emergence of HTML5 provides an alternative to these traditional approaches for collecting location data: create a smart phone web app. The recent adoption of support for HTML5 and associated technology such as Application Cache (AppCache), IndexedDB, and Geo Location in mobile web browsers now enables a relatively attractive alternative to native apps to reduce barriers to collecting geographic locations for points of interest into enterprise data systems.

AppCache allows web applications to declare files that should be cached by the browser to enable the application to load faster and to even run offline. LocationSharpener uses AppCache for operating offline. IndexedDB allows for local indexed storage of structured data from a web app. IndexedDB is a key-value database that allows the web app developer to specify indexes for optimized data access. Database size is typically capped at 5 MB. W3C Geolocation, associated with HTML5, is a standardized application

[1] PhoneGap, http://phonegap.com/
[2] mobl, http://www.mobl-lang.org/

programming interface (API) for accessing geographic location from JavaScript in a supporting browser on GPS enabled devices. When JavaScript requests Geolocation, the browser prompts the user whether they would like to share their location with this website. The functionality is contingent upon the user agreeing to share location data with a particular website. The evolution of mobile web browsers to support features of HTML5 and related specifications makes mobile web apps an appealing alternative solution platform for capturing geolocations in the field and later synchronizing this data with enterprise data systems.

3. PREVIOUS WORK

3.1 GPS Device

One common solution is to write a Geographic Information System (GIS) software application to run on a mid- or high-end GPS device using applications like ArcGIS Mobile. Another option is to write apps that run on GPS enabled mobile devices like smartphones. Both solutions require somewhat specialized skill sets or specialized hardware to implement, but offer advantages like making enterprise data available in the field for reference or for disconnected edit.

ESRI, a major vendor of GIS software and GIS software components, makes a product called ArcGIS Mobile which provides a range of options for offline data access and field data capture [5]. ArcGIS Mobile is also customizable to suit targeted business needs, and some companies specialized in creating custom solutions around it. Some of these apps even leverage HTML5, although these still rely on PhoneGap.

In perhaps the most closely related work to our own, the notion of using an HTML5 web app with offline data storage and use of Geolocation API was mentioned at a 2011 ESRI conference by presenter Adam Conner of Geodecisions [4]. However, the demonstrated application appears to have used geographic location as a parameter for centering the map on the user's current location to show nearby resources (fire hydrants in this case). By contrast, our solution collects and stores geolocation as primary data attributes of resources and utilizes offline data storage as temporary cache until the resource coordinates can be securely uploaded to cloud-based enterprise data system. When the user clicks a button, the app will collect the phone's geolocation which will be used to update the position of a well or river diversion in a cloud-based enterprise GIS enabled database.

Offline field GPS data collection is relatively common with native apps, but we were able to find no evidence that the industry has considered HTML5 web apps for GPS mobile field data collection yet, offline or otherwise. Companies like Mobenzi[3] and freeance[4] make customizable (native app) solution for GIS field data collection. Smartphone-based scientific field data collection of GPS and photographic information in a manner somewhat similar to that proposed here was explored by others, but it was implemented using a native app [3]. WebMapSolutions shared extensive customer feedback from 2011 that showed its clients want an easy way to collect GPS data in the field [6] but they are focused on solutions that use a native app (ArcPad, ArcGIS Mobile.)

3.2 Mobile App and Mobile Web App

Most mobile platforms in use today can run browsers capable of supporting many features of HTML5. Developers may see that it makes more sense to build an HTML5 web app instead of a native app for many scenarios. PhoneGap and mobl are just two of a large number of solutions to mobile platform segmentation. These allow developers to write an app once and deploy it as a native app to all major mobile platforms. While this solves the issue of developing separate apps for each platform, it does not address the problem of still having to install a native app on the device. Speaking about the sub-optimal state of this solution, one blogger at PhoneGap stated that one of PhoneGap's goals is to cease to exist (in favor of web apps running in fully-capable mobile web browsers) [12].

With respect to location data, the notion of utilizing client geographic location to customize web media has been around for at least a decade and in the last few years HTML5 and related technologies have significantly extended standards-based support for a large number of useful features that promote the web as a first class application platform [13, 14].

4. Mobile Web App Solution Design

We propose a platform-independent mobile web application that simply runs in an HTML5 capable browser. In contrast to solutions like PhoneGap and mobl, no native app needs to be compiled or installed since this just runs in a mobile web browser. We leverage mobile web browser geolocation support and HTML5 offline features like AppCache, Geo Location, and IndexedDB to simplify the offline collection and automatic upload of geolocation for points of interest that may fall outside cellular data coverage areas.

For this purpose, we built a cloud-based mobile web app called 'LocationSharpener' for accessing a disconnected dataset of water resource features loaded from an online data system and then cached locally on the mobile client. The user of the mobile web app clicks a button to capture and store the device's geolocation along with other updated attributes from the disconnected dataset, and later securely synchronizes that data to the online data system on the cloud (via REST-ful web services) when an internet connection becomes available. The web app uses HTML5 and JavaScript on the client tier and Java technologies on the cloud such as Spring Model-View-Controller (MVC) Framework and REST-ful web services, and Google App Engine for Java (GAE/J).

4.1 User Scenario

The user is an agency staff person, a well driller, a field scientist, property owner, or other person who needs to do a site visit to a well or river diversion and who is interested in making sure that the public records for that feature are as accurate as possible. Using an HTML5-capable browser on a smartphone, the user accesses the LocationSharpener website via HTTPS. The web browser downloads the files needed to later run in a disconnected state. Optionally, the user securely downloads offline records using tabular or map based query to later access and update from the field.

The user travels to the field location where their smartphone may not have any data access to the internet, and opens the LocationSharpener web app in the phone's browser (which loads the app from the previously cached files). The user selects the record that they want to update, and clicks the button "Use current location" to attach the phone's current geolocation to that record. The browser prompts the user to confirm sharing the current location with the web app. If the user grants permission, the app saves the location data to the local database. Later, when the phone's internet connection is restored, the app securely uploads the new location to the online web app (REST-ful web service) which updates the cloud-based GIS database.

[3] Mobenzi, http://www.mobenzi.com/
[4] Freeance, http://www.freeance.com/

4.2 Design

Figure 1 illustrates the sequential flow of the system to support the user scenario described in the previous subsection. The numbered list that follows describes the sequence of numbered events that are labeled in the figure:

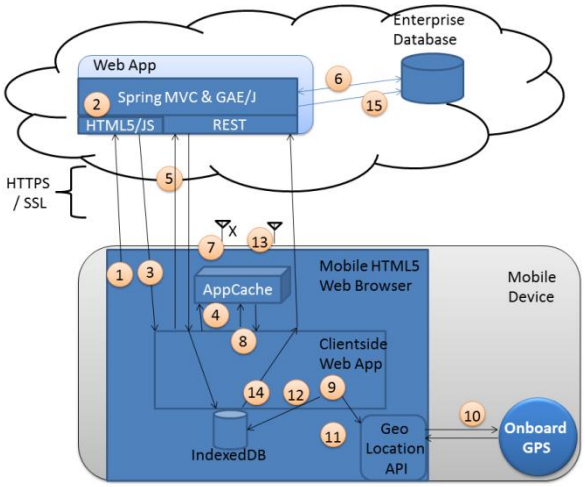


Fig 1: The sequential operation of LocationSharpener's primary components

1) While an Internet connection is available, the user accesses LocationSharpener website from a GPS enabled mobile device (such as a smartphone) with an HTML5 web browser.

2) A web server on the cloud responds with the various content files (HTML5, JavaScript, Cascading Style Sheet (CSS), and supporting images).

3) The client receives the content files and the browser of the client processes the files.

4) The content files direct the browser to persist files in the AppCache (part of HTML5) for offline browsing later.

5) A user accesses a page in the web app for choosing records to download prior to a field visit. The client-side web app sends REST-ful query to the cloud-based web server.

6) The web server retrieves the records from the enterprise relational database and sends the results back to the client as JavaScript Object Notation (JSON). The client side web app stores this data in IndexedDB (part of HTML5).

7) The user goes out into the field to visit the water resources infrastructure. Since this might be a remote area without reliable data coverage, the data signal could be lost.

8) Having arrived at the destination, the user accesses the client side web app. The browser loads the app from the AppCache.

9) The user brings up the record for a water well or diversion, which the client side web app loads from the local IndexedDB. The user clicks a button "Link to current location," and the web app calls the Geolocation API (an auxiliary feature associated with HTML5).

10) The Geolocation API acquires geographic position coordinates from the onboard GPS device.

11) The Geolocation API returns location to the client side web app.

12) The client side web app stores the location and metadata (such as time collected) back in the IndexedDB.

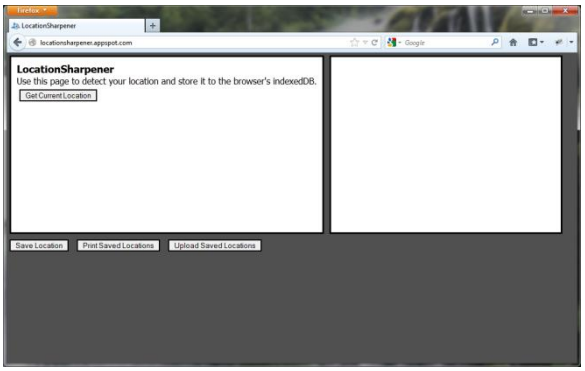
13) The user leaves the field site and eventually comes back within range of data connection.

14) The client side web app detects that a connection is available and uploads the updated location data through the REST-ful service on the cloud-based web server.

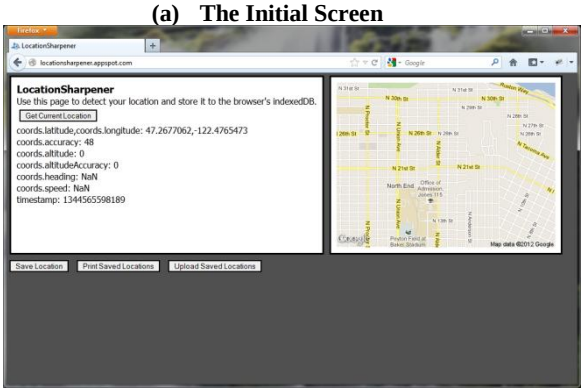
15) The server side web app saves the updated location data to the enterprise database.

4.3 Implementation

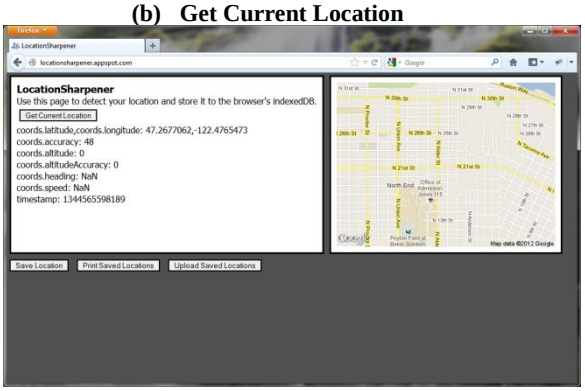
The solution is hosted in Google App Engine at http://locationsharpener.appspot.com. You can try it yourself if you have Firefox version 17.0 (recommended) from a desktop, tablet, or mobile phone. The application uses JSON over REST-ful services to download data from the cloud. Its demonstration is shown in Figure 2.



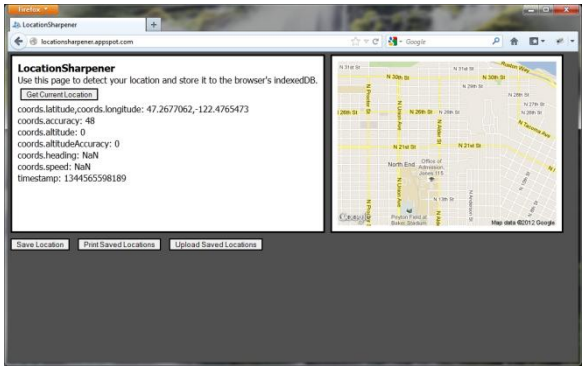
(a) The Initial Screen



(b) Get Current Location



(c) Save Location



(d) Upload Saved Location

Fig2: Demonstration of LocationSharpener

5. LOCATION ACCURACY ANALYSIS

To get an idea of the accuracy that could be expected from LocationSharpener, especially when a mobile phone was outside a service area and hence relying on satellite GPS without the benefit of Assisted GPS (aGPS), we brought the mobile phone and a handheld GPS unit into a semi-remote mountainous region and recorded a point on each device. We later calculated the distance between the two points in each pair.

The Android mobile device was an HTC Thunderbolt with Android 2.3.4 and Mozilla Firefox 17. The control device was a Garmin Oregon 450 handheld GPS with high-accuracy GPS. The test was performed at 35 locations in and around Mount Rainier National Park, Washington, U.S.A. Typical sites had view of the sky but not the horizons. Weather was overcast and raining. The error was calculated as the distance between the control point recorded by the consumer-grade GPS unit and the test point recorded by the smart phone. Average error was 23.3 meters with standard deviation 17.2 meters. This level of location accuracy is a dramatic improvement over the existing database locations in Washington State's well database, where typical error is about 500 meters. Our results show approximately 20 times improvement in accuracy over legacy data, even without cellular service. Table 1 shows our discoveries.

Table 1. A Summary of Accuracy Analysis

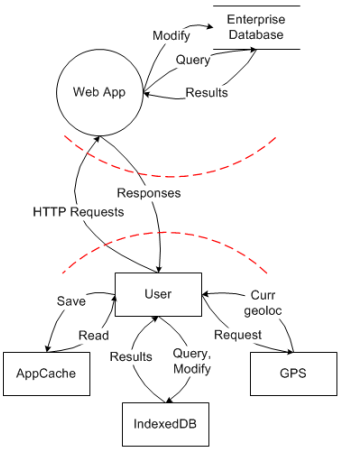
	GPS Devices	Mobile App	Mobile Web App
Platform Independence	No	No	Yes
Disconnected/Offline State	Yes	Yes	Yes with only HTML5
Used for the WA State Geolocation DB	Yes	No	No
Location Accuracy (error ratio, compared to the WA State Geolocation DB)	Low (500 meters)	Not Tested	Higher (23.3 meters)

6. THREAT MODEL ANALYSIS

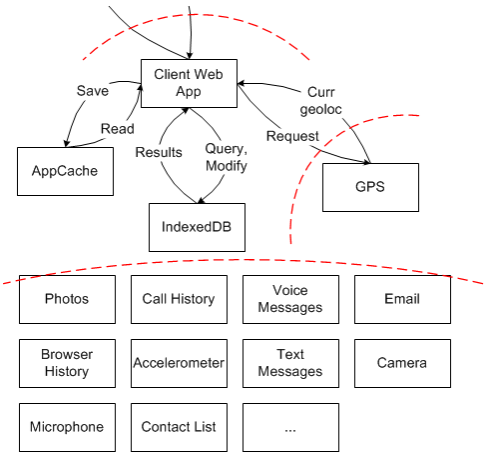
We constructed a threat model of the LocationSharpener using the Microsoft Security Development Lifecycle (SDL) Threat Modeling Tool [9]. We used this tool to create a diagram of the system (see Figure 3), and then used it to analyze the model for security threats. The tool lists security threats by threat categories, which are Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege - collectively abbreviated STRIDE. An example of a threat identified by the tool is tampering with the HTTP request from the user to the web server. A partial mitigation for this is to use strong encryption via HTTPS (where HTTP messages are sent over Secure Socket Layer (SSL), which encrypts the messages.)

The Threat Modeling Tool provides fields for the developer to document the impact of each threat type and how the threat is mitigated by the system. The tool provides contextual information for the threat types and suggestions for mitigation, making it a helpful tool for systematically analyzing and mitigating the security threats to a system.

Aside from this conventional threat model which identifies threats to the application and its components, we constructed another threat model that specifically looks at threats to user privacy from an HTML5 app (Figure3a) as compared to a native app (Figure 3b). Browsers provide a "sandbox," or special, restricted execution environment, in which JavaScript code is run. One function of the sandbox is to prevent web sites from accessing private resources (e.g. user's personal data) without permission. (A threat model that specifically addresses the security threats that arise from the new features of HTML5 is outside the scope of this paper.)



a) Mobile Web App



b) Mobile App

Fig 3: Threat modeling of a Mobile Web App and a Mobile App

The degree to which web apps are sandboxed by browsers is one of the traits that distinguish our mobile web app solution from a native app. The extent to which a web app can interact with the device is far more limited than a native app. The decreased access of a web app to personal data compared with a native app, which is summarized in Table 2, is one important factor for users to prefer a web app rather than a native app [7].

Table 2. A Summary of Threat Risk Modeling Analysis

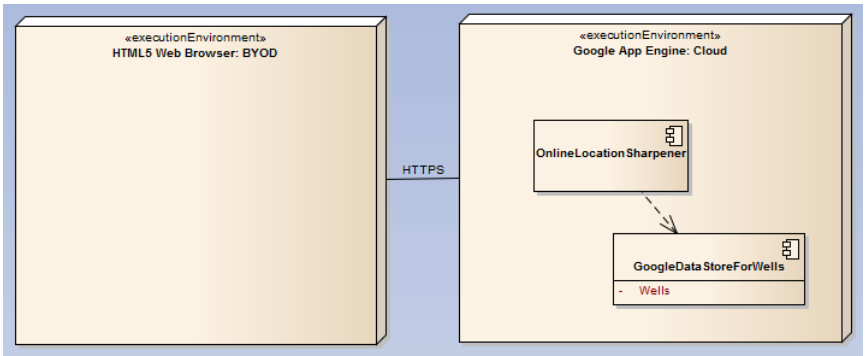
	GPS Devices	Mobile App	Mobile Web App
The number of trust boundaries	0	2 + *n*	2
Vulnerabilities	Very Low	High	Low

n: the number of mobile apps

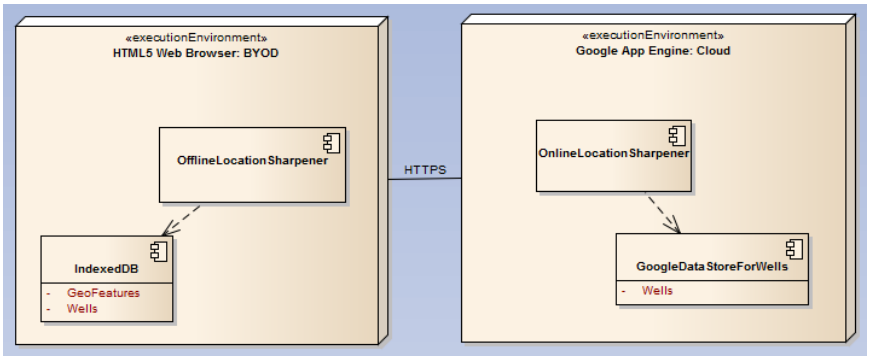
Beside sandboxing, it is a necessary constraint that a web app running in a browser must have less than or equal access to private data on the device as a native app, since the web browser itself is merely a native app. The user privacy threat model helps us enumerate the additional threats posed by a native app versus an HTML5 web app. Each application programming interface (API) to a module on the device, such as GPS, provides information flows across a trust boundary (shown in the diagrams as red dashed curves.) Each method exposed by the API is one type of information flow. An HTML5 mobile web app accesses far fewer APIs than native apps, and fewer methods per API. For example, in the case of GPS, there is only one method for HTML5 versus 27 methods in the Android 'LocationManager' class that is exposed to native apps.

7. ARCHITECTURE ANALYSIS

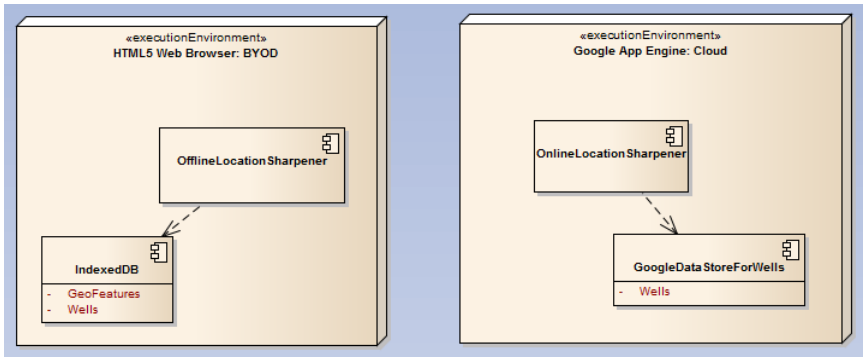
By using architecture modeling for the development of the mobile web app for offline or online data collection, we discover guidelines for prospective developers of mobile apps



(a) t=0: Mobile HTML5 Web browser requests content from the cloud for the first time



(b) t=1: Browser stores downloaded resources locally.



(C) t=2: Browser stores downloaded resources locally

Fig 4: UML Deployment Diagram at t

to use when they are considering whether to use a browser based HTML5 app. We created a visual model by using a software architecture documentation methodology - 5W1H Re-Doc [1]. 5W1H Re-Doc is a methodology that guides a software architect to document architecture in terms of 4+1 view model [10]. LocationSharpener is represented as a series of views, each view highlighting a different perspective of the software system - such as use case, logical design, message exchange process, component, and deployment views. For example, we created a series of three UML deployment diagrams rather than the traditional single deployment diagram that was sued for a web application using HMTL4. Our diagrams, shown in Figure 4, show three deployment modes at the following times, which were not necessary in a mobile app:

- At *t*=0 (Figure 4a), the cloud portion of the app has been deployed to the Google cloud, Google App Engine. Before visiting the website for the first time, the client (HTML5 web browser on a mobile device) does not have the client side files deployed to it yet.
- At *t*=1 (Figure 4b), when the user accesses the website and grants permission for the site to store offline files, the files are downloaded to the client and stored in the AppCache. The offline web app is now "deployed" to the client.
- At *t*=2 (Figure 4c), when the user goes offline, we display a diagram showing the app deployed to the client and the cloud but without the connection between the two. The client app can run independently of the cloud in offline mode. When the connection is later restored, Figure 5 (*t*=1) again illustrates the state of the system. The client can go online and offline indefinitely, cycling between Figures 5 and 6 (*t*=1 and *t*=2.)

By creating this 4+1 view architectural model of our HTML5 web app, we learned that the ability of HTML5 websites to run in a disconnected mode via AppCache has some special implications for the software architecture. AppCache is a new feature in HTML5 that lets a web site declares a set of files that should be downloaded by the web browser and saved on the client's machine. Subsequently, when one of these files is needed, the browser accesses the local copy rather than downloading it again from the web server. This can be done to speed up performance, reduce network traffic, or (in our case) to allow certain functionality from the website to be used when the browser is not connected to the Internet. Because of that, the Mobile Web App developers need to consider two subsystem – online and offline subsystems – at the beginning of designing the app, which is shown in Figure 5.

Table 3 shows comparisons of GPS devices, mobile app, and mobile web app solutions for collecting geolocation field data in terms of the distributed computing paradigm, the cases of deployment modes, and the minimal number of subsystems.

Table 3.A Summary of Architecture Analysis

	GPS Devices	Mobile App	Mobile Web App
Distributed Geolocation Application	No	No	Yes
The cases of deployment modes	1	1	3
The minimal number of subsystems	1	1	2

8. CONCLUSION AND FUTURE WORK

A mobile web app using HTML5 technology brings several benefits to mobile users and developers. First of all, through the demonstration of the mobile web app 'LocationSharpener', we showed how we leveraged the new HTML5 features such as IndexedDB, AppCache, and GeoLocation to simplify the offline collection and automatic upload of geolocation for points of interest that may fall outside cellular data coverage areas. These features allow a mobile web app to be a distributed client/server system that works in a disconnected state. IndexedDB worked without errors although testing the data integrity or validity of the feature was not within the scope of our tests. It recorded and reported back the data consistently without corruption, error, or software exception. AppCache generally worked, but the offline files seemed to be retained by the browser even after we changed the browser's site settings to disallow saving offline content for our site. We re-loaded the page, cleared the browser's site settings, updated the server-side files, reloaded the page again, and this time confirmed that we would allow the site to store offline content. The Geolocation API consistently allowed our web app to obtain location data

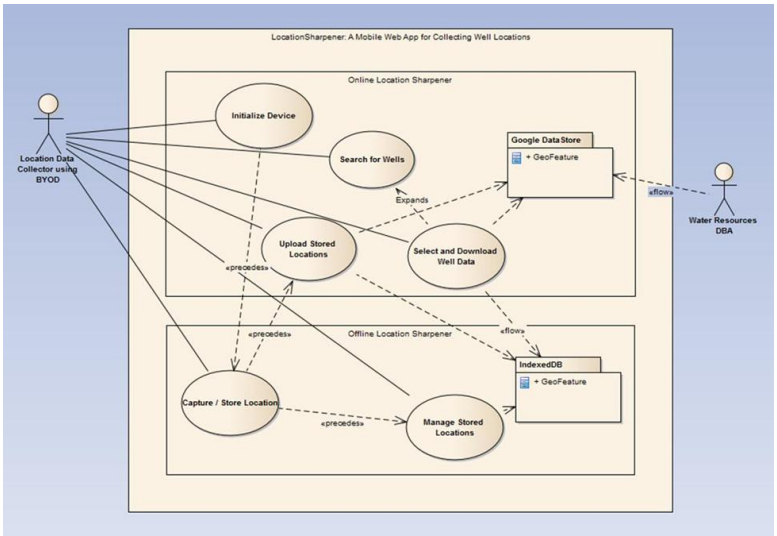


Fig 5: UML Use Case Diagram showing two subsystems – online and offline.

from the phone after the browser prompted the user to confirm that they wished to share their location with our site.

Secondly, the analysis of location accuracy showed the HTML5 mobile web app approach provides acceptable location accuracy even when offline. We collected locations of water resources facilities like wells and water diversions and analyzed how accurately the mobile web app collects the geolocation data. Out test results showed that the level of location accuracy was a dramatic improvement over the existing locations in Washington State's well database. The HTML5 mobile web app demonstrates that we can employ the concept of a smart phone's BYOD (Bring Your Own Device). The cost of specialized data collection devices can be saved and persons such as property owners, drillers, scientists, and the general public can be empowered to collect geolocations.

Thirdly, in contrast to native mobile apps, mobile web apps offer two clear advantages to users and developers in terms of higher platform independence and lower vulnerability: one code base allows us to deploy our platform-dependent mobile app onto multiple mobile platforms equipped with a HTML5 compliant web browser, which shows the concepts of "Write Once Run Anywhere" [2]. Also, we conducted threat risk modeling and analysis and visualized the trust boundaries comparing a mobile native app and a mobile web app. The HTML5 mobile web app has far fewer trust boundaries to be secured.

Lastly, in the process of building this mobile web app for offline or online data collection, we gained some experiences that we can pass along to other mobile app developers to help them make an informed decision of how to design a mobile web app. The documented software architectures in 4+1 views show that the mobile web app developers need to consider two subsystems in terms of online and offline in the use case view. For each subsystem of a mobile web app, the mobile web app developer needs to design and implement class hierarchies, message exchanges, and physical components. Also, the developer must be aware of the deployed subsystems having connected, disconnected, and connected communication sequences over time. Those practices have not been required for developing native mobile or mobile web apps without using HTML5.

9. REFERENCES

[1] Sam Chung, Daehee Won, Seung-Ho Baeg, Sangdeok Park, Service-Oriented Reverse Reengineering: 5W1H Model-Driven Re-Documentation and Candidate Services Identification, In Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA'09) December 14-15, 2009, Taipei, Taiwan.

[2] ComputerWeekly.com. 2013, Write once, run anywhere? http://www.computerweekly.com/feature/Write-once-run-anywhere, Accessed on March, 29, 2013.

[3] Luis Corral, Alberto Sillitti, Giancarlo Succi, Alessandro Garibbo, and Paolo Ramella, 2011, "Evolution of mobile software development from platform-specific to web-based multiplatform paradigm," Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software (Onward! 2011), October 22 - 27, 2011, Portland, OR, USA, Pages 181-183.

[4] ESRI, 2011, Dev Meet Up – Philadelphia, PA. http://blogs.esri.com/esri/arcgis/2011/06/05/dev-meet-up-philadelphia-pa/. Accessed on March 27, 2013.

[5] ESRI, 2013, ArcGIS for Smartphones and Tablets, http://www.esri.com/software/arcgis/smartphones/demos.html, Accessed on March 5, 2013.

[6] ZefHemel and EelcoVisser, 2011, "Declaratively programming the mobile web with Mobl," Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications (OOPSLA 2011[, October 22-27, 2011, Portland, Oregon, USA.

[7] Jan Lauren Boyles, Aaron Smith, and Mary Madden, 2012, Privacy and data management on mobile devices, Pew Internet & American Life Project, September 5, 2012, http://pewinternet.org/Reports/2012/Mobile-Privacy.aspx, accessed on November 25, 2012.

[8] Qing Li and Greg Clark, 2013, Mobile Security: A Look Ahead, IEEE Security & Privacy, January/February 2013, Volume 11, Issue 1, pp. 78-81.

[9] Steve Lipner, 2004, The trustworthy computing security development lifecycle, Computer Security Applications Conference, 2004. 20th Annual, vol., no., pp. 2- 13, 6-10 Dec. 2004.

[10] Philippe B. Kruchten, 1995, The 4+1 View Model of architecture, Software, IEEE, vol.12, no.6, pp.42-50, November, 1995.

[11] Oregon Water & Monitoring Well Data Standard Draft 1, http://apps.wrd.state.or.us/apps/gw/owmwds_well_query/docs/ORWaterWellDataExchangeStandard_v1Draft.doc, Accessed on March 5, 2013.

[12] PhoneGap, 2013, PhoneGap Beliefs, Goals, and Philosophy. http://phonegap.com/2012/05/09/phonegap-beliefs-goals-and-philosophy/. Accessed on March 27, 2013.

[13] Antero Taivalsaari, TommiMikkonen, Dan Ingalls, and Krzysztof Palacz, 2008, Web Browser as an Application Platform," the proceedings of the 34th Euromicro Conference Software Engineering and Advanced Applications (SEAA'08), 2008 , Pages: 293 – 302.

[14] Antero Taivalsaari, TommiMikkonen, MattiAnttonen, and ArtoSalminen, 2011, The Death of Binary Software: End User Software Moves to the Web, 2011 Ninth International Conference on Creating, Connecting and Collaborating through Computing, 2011 , Pages: 17 – 23.

[15] Kerry Taylor, 2006, The semantics of water, In Proceedings of the Second Australasian Workshop on Advances in Ontologies - Volume 72 (AOW '06), Mehmet A. Orgun and Thomas Meyer (Eds.), Vol. 72. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 3-3.

[16] Washington Department of Ecology Water Resources Web Map, 2012, http://www.ecy.wa.gov/programs/wr/info/webmap.html. Last accessed 10/20/2012.

[17] W3C, 2011, HTML5 differences from HTML4, http://www.w3.org/TR/2011/WD-html5-diff-20110405/, accessed on March 27, 2013.