Meta-App A Pull-based Approach

Tushar Badgu
Department of Computer Engineering
Pune Institute of Computer Technology, Pune

Omkar Patil
Department of Computer Engineering
Pune Institute of Computer Technology, Pune

Abhidnya Patil, Snehal Rasakar
Department of Computer Engineering
Pune Institute of Computer Technology, Pune

ABSTRACT
Now-a-days Smartphone has become common and are used by millions of people. As these Smartphone provide a facility of custom built apps, many of the businesses are moving towards building apps. A lot of time and capital is being invested in developing these kinds of apps. However in order to reach out customers these apps are being built hap hazardously. Poorly written apps pose the risk of exposing user data and breaching privacy. Development and maintenance of these apps is also required. Furthermore some apps are being built only because everyone is moving app-only. All this has made it difficult for a Smartphone user to keep track of different apps on his devices. This paper aims at creating a Meta-App which would act as supplant to all the redundant applications and will be based on intent publishing. A solution to provide fine control to the user over his data that is shared with the service providers and implement a Pull-based Approach.

General Terms
Content Delivery Systems, User Privacy, Intent Publishing, Security.

Keywords
Public key cryptosystems, standards, Query Processing, Content Analysis and Indexing, Commercial services, Data sharing, Natural language interfaces, Language parsing and understanding, Text analysis

1. INTRODUCTION
Nowadays, we can find an android application developed by every other service provider to enable the user's easy access to their products simply by a click of a button on their phones. However, this necessitates the users to download the applications provided by each of these providers. Thus the user spends a lot of time compare the prices and offers by looking into each of these applications and find the best available deal. Also many of these applications are poorly written and thus exploit unnecessary personal information of the user [1].

For example, Amazon , Flipkart , Myntra are all leading providers in providing products ranging from home appliances to electronic items , from clothing to footwear and even super market products. Uber, Ola, Meru are all the providers of taxi services to the user .Similarly, make my trip, yatra.com, travel advisor are useful when planning a trip and includes everything from transport bookings and hotel reservations. If a user thinks of buying shoes, he tends to check the prices offered by each of the providers to get the cheapest deal for him. Keeping this in mind, we design a tool which just asks the user to specify his search. The service providers will be provided with API's which can be used to publish their services to the users. We will be developing an efficient ranking strategy to order the search. And the best possible deal will be provided to the user without the need of installing so many apps and without even compromising the security of his personal information. A security layer will be developed over normal android security to give the user a complete control of sharing of his personal data.

Also everyone is moving app only so there is trend in today's market for having app for their product [2]. Development and maintenance of these apps is also a major problem only few people are concern about it. Some apps required basic information of users such as name, email, phone number, etc. which is not required actually. Because of this app going trend everyone is developing app but no one is concern about fine UI, security and privacy so this poorly written apps harms the operating system. That is where this Meta-App comes. It is basically considers the need of security and privacy of user while using app. If we are going to ask for some product, the server will give us information about various products on various apps with the help of various APIs so it is kind of pull based approach.

A person in the future may only just check Meta-App to check which provider giving him fast and best services. Some features might achieve this implicitly and some might be explicitly built for the same.

2. BACKGROUND
When a user installs or uses any free app there is a strong possibility that it may send sensitive user data to third parties [3], [4]. As these apps are very popular on Smartphone, app users worry about the amount of personal information shared by the app. In a survey by Pew Research Centre regarding this topic it was found that 54% of the people refused to install a particular app after they found out how much personal information they needed to share to use it. Also 30% opted to uninstall the app when they learnt that it shared information they did not want to share.

Why this concerns the user?

- One of the reasons is that it may share unique ID's related to devices like IMEI, MAC address, IMSI, etc.
- An app can also try to access device functions and personal data by requesting user permission for access to network, phone calls, location or even at times hardware control. this can lead to over privileging (access more data than required for advertising and data collection)
- Third parties like advertisers can be sold this data collected by the app.

Wall Street Journal (WSJ) conducted a survey of 101 popular Android and iOS apps in 2010 and examined the data they transmitted. It found that 56 sent device's unique ID, 47 sent

device location, 5 sent age, gender and other personal details without user's awareness or consent.

Given these conditions we tried to study the kind of personal data shared by the apps and to whom. There are three approaches to survey the data shared by apps

- Permission analysis

  It involves the analysis of the permission requirements disclosed to user before app installation or use. It helps review thousands of app efficiently but the review is only 'high-level' i.e. we can't know whether the app actually collects the data it requests permission for. One study found a correlation between the numbers of downloads and the number of permissions requested. Also a number of "look-alike" apps to popular apps request more permissions that the original app [5].

- Static code analysis
  It involved decompiling an app to identify the data it sends by understanding the design .it provides more detailed insight and is relatively easy to automate. In one review of 114,000 apps it was found that on an average every app had at least one ad library which requested access to services like Wi-Fi network, Camera, SD card, etc. It can also be used to find over privileging and uncover potential malware and adware [6].
- Dynamic analysis
  It involves capturing the data transmitted when an app is actually used. In one study, it found 97 out of the 145 apps tested sent potentially private information such as phone information, device IDs, or geo-coordinates to primary or third-party servers. We can also use a Virtual Private Network (VPN) to monitor the traffic from the device.

In a survey [7], 110 free apps were tested of various categories like Business, Games, health & fitness, etc. Many mobile apps transmitted potentially sensitive user data to third-party domains, especially a user's current location, email, and name. In general, iOS apps were less likely to share sensitive data of nearly every type with third-party domains than were Android apps, except for location data .The average Android app sent sensitive data to 3.1 third-party domains, and the average iOS app connected to 2.6 third-party domains. The top domains that received sensitive data from the most apps belonged to Google and Apple.

3. EXISTING SYSTEMS

There are two approaches which have been explored and are used currently:

1) Mobile Native Apps
A mobile native app is built keeping in consideration a particular device and its operating system. These native apps are downloaded from a web store and installed on the device in contrast to a web-app that is directly accessed directly over the internet.

The disadvantages of using native apps are as follows:

- Publishers are usually not willing to share information about their subscribers to the app store which is a necessity if they intend to deploy their app on the store.
- The publishers have to deploy their apps for a wide range of devices. Separate versions of the app need to be developed, and tested for different environments like Android, IOS, and Windows. Along with the development cost this introduces additional maintenance costs. Even though cross platform frameworks have eased the efforts needed for such development, it usually compromises on the time factor and ease of development.
- For updating these applications, different teams need to be allocated for making necessary changes for every platform the app is deployed on. Not only does this increase the amount of work but also causes a bit of inconsistency in the working of apps on different platforms.

2) Mobile Web Apps/Website based approach
A mobile web app is a web application accessed through the web browsers of mobile devices like tablets, Smartphone. It is built with three core technologies:- HTML, CSS and JavaScript. Unlike the mobile apps mentioned above, these websites are independent of the platform and devices which it needs to be run on.

The disadvantages are as follows:

- Mobile browsers have less functionalities as compared to traditional desktop browsers. It is usually similar for the more popular Platforms but it varies greatly for other ones. So the web apps are developed such that they support the lowest common framework in order to make it available to every user. This however gives the website a clumsy look. If this has to be avoided, the publishers will have to customize these web apps for various browser version which is time consuming.
- Web apps generally cannot access the mobile hardware and software. The use cases which require the web apps to access mobile hardware like camera control, GPS control, and onboard sensors are directly ruled out. Because of this limitation, development of complex graphics such as motion control in gaming cannot be supported.
- The running of web app requires a continuous web connection to function unlike mobile apps. Thus, the cost of accessing these web apps requires to be paid to the network operators. Thus costs can be completely avoided in case of mobile apps which requires no internet connectivity which poses to be a great disadvantage in case of web apps.

4. PROPOSED SOLUTION

The proposed solution in this paper is an entire new Framework based on Pull-based approach rather than existing Push-based approach. This framework can be thought of as a service which will forward user intents to sellers and their responses back to user [8].

4.1.Architecture

The system consists of four modules:-

1) User application 2) Server 3) Multiple proxies 4) Sellers SDK

4.1.1. User application.

This component will be deployed on the user side. It can be a web app or any android, IOS, windows app. It will act as the point of communication from the user perspective. For a user to use this service he must first register to it and perform an initial setup Installation and registration. The framework uses a multi-level encryption mechanism to transfer data between the entities. It will use a two way public-private key encryption algorithm based on TLS. During writing the client application the public key of the service server will be saved in it. The demo application uses an RSA algorithm to generate this key.

Whenever the user registers on the application it will perform following:-

1) The server will have its own pair of RSA public-private key. The server public key will be saved in the app to encrypt all outgoing data from the app to server.
2) Whenever a user registers he will get his unique UUID from the server. Along with it an RSA algorithm will run on the app to generate a user specific Public-Private key.
3) The User-Public key will be stored in the server to create a downlink encryption channel.
4) However time required for data encryption and decryption is large for RSA. So we propose a session channel.
5) We propose creating session for data exchange of fixed time-frame which can be decided by the user. RSA will be used to manage establishment of these session.
6) The client public key and UUID will be given to an AES algorithm to generate a cipher text.
7) The seed value of AES which will be used to generate the cipher text will then be encrypted using server-public key to generate a cipher-seed and sent to server.
8) The server can then decrypt the seed value from the cipher-seed using server-private key and use it to decrypt the data sent by the client.
9) This seed-value will vary from session to session and would thus reduce overheads.
10) The UUID generated will be used for identifying user requests and categorization of the same.

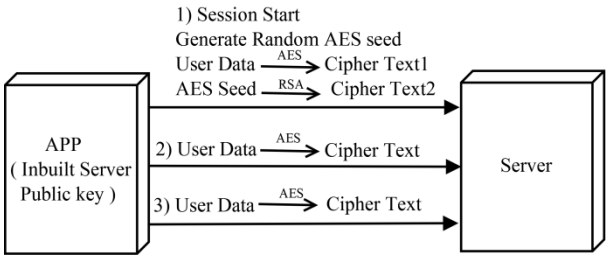

Fig 1. Communication Module

Publishing intent:

Whenever user wants to publish intent, he will browse through the app. The services will be listed in the form of tree which will contain multiple children containing sub-categories. When user selects a particular service intent will be generated. The selected service which user wants to avail will be then divided into topic and subtopic. For example, if user asks for Indian wear in clothing category, the query will get separated into topic as clothing and subtopic as Indian. The necessity of using such sub-division is because ZMQ used fixed string topic filter, however the subtopic filtering will be done using our own APIs. The problem faced in using ZMQ filtering for the whole content is that the order in which the query arrives should match with the format the service provider specifies, if not matched the interested subscriber won't get the relevant queries only because of an inconsistency in the order. For example, the service provider specifies the topic filter as traditional clothing however our topic filter specifies clothing traditional, there will be a discrepancy and the purpose of the filter will not be served.

This intent will consist of the UUID and a request-id which will be unique for each request made by the user. The combination of UUID and request-id will be used to uniquely identify the request on the server and proxies. The intent will then be encrypted using the server public key and sent to the server. The message will be of JSON format with the keys being as follows:

UUID: Unique user identity which will be provided to the every user when he registers.

Request-id: request-id is a unique id provided for every request.

Deadline: deadline is the time in which request should process.

Subtopic: Subtopics are the more detail information of the category of topic.

Description: The user will describe his needs, specification he wants as well as subscriber information.

Displaying the responses:

Whenever a seller response is received the app will first decrypt it using the users own private key as the data it receives will be encrypted by users public key. It will then be stored locally on the device and wait for users' action. A user can decide what he wants to do then.

Subscriber Namespaces:

The user can regulate the data which will be available to the subscriber using these namespaces. User will be given an option to decide which of his data he wants to make public to a subscriber. Only this data can be accessed by the subscriber through the service. Additionally the user may also be provided functionality to create multiple namespaces for different subscribers. This facility can be used in a variety of services like a taxi service should only know your current location or the location you intend to have the taxi. Another example can be of an e-commerce service which should only be able to access those text messages having certain extensions like VZ, IM, etc. and not the entire message log.

4.1.2. Server.

The server will perform the following tasks

1) Updating user databases

Whenever a new user registers a UUID will be generated and this UUID will act as an index to locate the details of user.

These details will include his username, password and the public key of that user which will be generated by the application when the user registers. This key is necessary to encrypt the data which will be sent to the app.

2) Managing Seller APIs

The sellers will be provided with APIs which they can implement to get the user requests. These APIs can be a set of SDK developed for the service. For a seller to implement these APIs we propose seller registration. All the sellers who want to implement the API's will need to register. This is specifically needed to prevent other sellers who have not registered for a specific service from getting offers of those services. Also it would help in ranking of the sellers in the next phase. When a seller registers he will specify the services he will provide. This list of services would be used during filtering to identify the topics for which the seller has subscribed. ZMQ doesn't provide any way of smartly filtering the data so this mechanism will ensure it.

3) Forwarding intents

The user intent received from the app will be decrypted using the encryption-decryption model and then will be broadcasted to the sellers using Publisher-Subscriber Framework along with the intent information. ZEROMQ distributed messaging framework is used to implement the communication between server and dealers. In particular, the Pub/Sub pattern is used of ZMQ framework. Publish/Subscribe is a classic pattern in which the sender entity is called publisher and the receiving entity is called subscriber. Unlike the usual networking communication, the publisher is not programmed to send the message to a particular subscriber. Instead, Messages are published without the knowledge of if there exists any subscriber interested in that particular topic or not. Messages published will have a topic set by the publishers which are then eventually used by the subscriber as a filter for topics of interest. Pub/Sub communication model is asynchronous and independent of when the publisher and subscriber is started. Hence, the subscriber will not receive the messages which the publisher had sent before the subscriber was started.

The Intent description provided by the user will be in textual form hence not much useful. So, extraction of metadata from the description is performed which can be queried by the seller in order to provide responses for the request. In the supervised approach, a maximum entropy (MAXENT) classifier is used to determine whether a unigram word is a keyword (binary classification). Each candidate word is represented by a variety of features such as TFIDF, Position features, Stop-word features, Sentence features, Lexical features, Summary features and speaker features. One way for model training and testing in this supervised approach is to simply use all the words as instances. In addition, we introduced a bigram expansion module which aims at extracting “entity bigrams” using Web resources.

4) Managing Proxies

The intent information received by the server will also be forwarded to the proxies for further management. Additionally it would also involve the load balancing of all the requests managed by the proxies.

4.1.3. Proxy.

This can be thought as the regular and anti-spamming filter of the service. Whenever the seller will put an offer to a user it will pass through this proxy. The proxy will check the validity of the Request-id. All the valid responses will then be grouped and ranked according to certain criteria. This will ensure only the best of the offers will reach the user. The criteria can be defined by the user or could use heuristics.

1) Invalidation of Offers

Any seller which was not sent a Request-id will be blocked from responding to the intent. Also a particular time limit will be issued to each request. If any response occurs after this time limit it will be blocked. Responses to completed intents will be blocked. The proxy will be instrumental in filtering the unwanted responses from all the responses.

2) Ranking

Ranking will involve assigning a benefit value to all the offers which will be received by the proxies from the sellers. These benefit value will then are used to grade the offers and filter out all the ineffective or non-beneficial offers.

The traditional approach for ranking was based on evaluation of specifics parameters like response time, security, price etc. based on public information released by service providers or test results from repeatedly invoking a service. However the authenticity of the information provided by sellers needs to be verified which is practically impossible. Also it lacks an effective way of identifying quality features that users are actually interested in when choosing a service [9].

One of the approaches for ranking can be based on First Come First Serve basis with a user feedback. Whenever a particular transaction is completed the users can rate the seller of the service based on parameters like time to complete, QOS, etc. This grading can be a basis for the ranking of the sellers. This technique can be effective in conditions where a seller is quick to respond but lacks the service; such a seller would be ranked lower. This will ensure that only those sellers who provide good service will have a chance to reach user if they reply a bit late but before the deadline.

Another approach for ranking is based on sentiment analysis of user reviews. The user reviews are first evaluated by ignoring the sentences in the reviews which are not related to the quality of service. Such sentences can just be a description of the service which has no relevance to the features of the service. The users’ reviews are then evaluated to find the interestingness of the feature by the users’ perspective. Some features are more popular among the users than others and the ranking should be based on such more popular users. Logistic Regression can be used to determine these impacts. Finally the sentiment analysis is done for ranking by classifying each feature as positive or negative and this classifier is learned through a supervised learning process [10].

Transferring offers in a secure way to the user it is particularly important that whatever offers are being sent to the user must be provided required security as it may contain sensitive data which may reveal user's identity or certain details. To avoid this same session mechanism user in server can also be used at the proxy. This would ensure that even if the messages are intercepted they will not be readable.

4.1.4. Sellers SDK

All the functionalities to be used by the sellers are exposed as an API by the server. Now to make use of such APIs, the seller needs to implement the functions provided within the SDK. Various functions that will be provided within the SDK will be

1) Registration of the seller

2) Fetch user intents

3) Send responses for the intents

4) Create a new namespace

5) Store data in the namespace

6) Fetch data from the namespace

4.2.Architecture Diagram

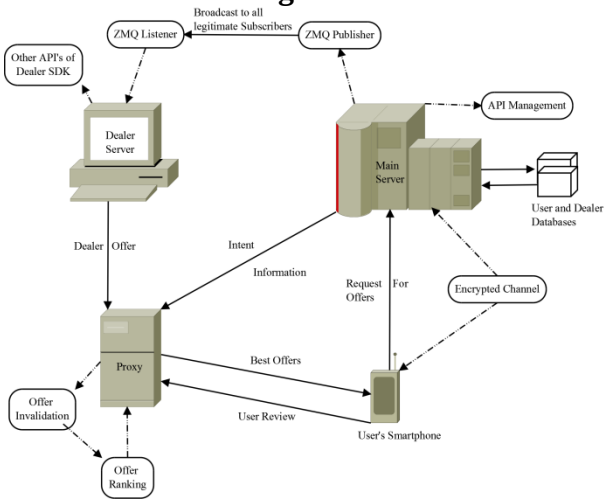

Fig 2. Architecture Diagram

4.3.Flow of System

Flow Diagram for the system is as follows:

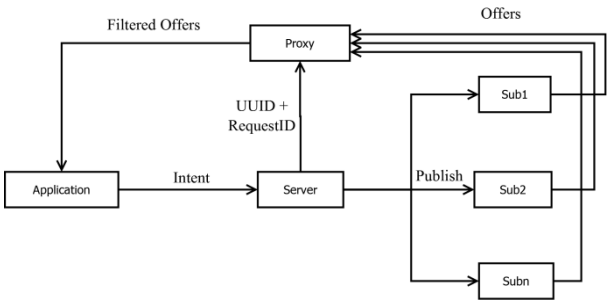

Fig 3. System Flow Diagram

5. EXPECTED OUTCOMES

The chances that users' sensitive data will get compromised is reduced by the fact that none of the data is being stored anywhere apart from the app itself. While registering for the service minimal amount of user information will be requested. A user will never be identified by any information which can be used to reach him in the future like his phone number. This framework can help reduce the need of user engagement type of services thus reducing the need for a user to install these types of applications on his phone. The seller APIs can provide means to go digital cheaply without investing in the development of an entire end-to-end application service.

6. CONCLUSION AND FUTURE WORK

Meta-App is a step towards making the user aware about their privacy and also protect it. Meta-App is a project with two fronts. One is to encounter the problem of everybody moving towards the App-only approach. To make use of any service the user has to install an application for that service provider and with this comes the exposing of user private data to outsiders for their own advantage. Thus Meta-App introduces the concept of One-App approach in which only a single app has the power of all such apps combined. The other front is to develop a mechanism to provide a pull-based notification service to users and also put the privacy of users' data in its own hands. The user has the power to choose what to share and what not.

Thus, we conclude by completing a thorough study of current systems it is necessary to design a system which can provide a more fine grained control to the user data and also provide a platform to the user to replace the most of the redundant applications on the phone. We would conclude that by doing this project we can put the next step in protecting user privacy.

7. ACKNOWLEDGEMENTS

We would also like to show our gratitude to Prof. Kalyani Waghmare, PICT and Mr. Abhijit Gadgil who provided insight and expertise that greatly assisted the research. We are immensely grateful to them for their comments on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

8. REFERENCES

[1] Taenam Cho, Seung-Hyun Seo, Vulnerabilities of Android Data Sharing and Malicious Application to Leaking Private Information, 2013.

[2] Nai-Wei Lo, Kuo-Hui Yeh, Leakage Detection and Risk Assessment on Privacy for Android Applications: LRPdroid, 2014.

[3] Hongliang Liang, Dongyang Wu, Jiuyun Xu, Hengtai Ma, Suvery on Privacy Protection of Android Devices, 2015.

[4] Kuo-Hui Yeh ,Nai-Wei Lo, Chuan-Yen Fan An Analysis Framework for Information Loss and Privacy Leakage on Android Applications,2014

[5] Muneer Ahmad Dar, Javed Parvez, Enhancing Security of Android & IOS by Implementing Need-Based Security (NBS), 2014.

[6] ZheMin Yang, Min Yang, LeakMiner: Detect information leakage on Android with static taint analysis, 2012.

[7] Zang J, Dummit K, Graves J, Lisker P, Sweeney L. Who Knows What About Me? A Survey of Behind the Scenes Personal Data Sharing to Third Parties by Mobile Apps. *Technology Science*. 2015103001. October 30, 2015. http://techscience.org/a/2015103001

[8] Thejovardhana S. Kote and S. R. Jeyashankher, A large scale publish subscribe platform for information delivery to mobile phones, 2006.

[9] Keke Chen, Ya Zhang, Zhaohui Zheng, Adapting Ranking functions to User Preference, 2008.

[10] Xumin Liu, Arpeet Kale, Javed Wasani, Extracting, Ranking, and Evaluating Quality features of web services through User Review Sentiment Analysis, 2015.