IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

Personal Cloudlets: Implementing a User-Centric Datastore with Privacy Aware Access Control for Cloud-based Data Platforms.

Nauman Birwadkar[1], Mrs. Monali P. Deshmukh[2], Apoorva Arbooj[3], Suraj Todkari[4], Priya Agarwal[5]

Professor, Department of Information Technology, JSPM's Rajarshi Shahu College Of Engineering, Pune[2]

Student, Department of Information Technology, JSPM's Rajarshi Shahu College Of Engineering, Pune[1,3,4,5]

Abstract: This paper presents OPENi's Personal Cloudlets framework as a novel approach to enhancing users control and privacy over their data on a data driven, cloud-based platform. We introduce the EU FP7 funded project OPENi, the OPENiconcept, and the research objectives that influenced the design and implementation of OPENi's Personal Cloudlet Framework. We outline OPENi's architecture and describe how through the use of REST based endpoints, object-based access control, OPENiTypes, and stateless JSON Web Token (JWT) it allows users share, reuse, and control access to their data across many mobile applications while maintaining cloud scalability. Furthermore we describe how a number of the Personal Cloudlet Framework's features enhance a users privacy and control, including: the User Dashboard, the Privacy Preserving Data Aggregator, and the fine grained access control mechanism.

Keywords: Personal Cloudlet, OPENi, Cloud Security, User Centric.

I. INTRODUCTION

A central aspect of the contemporary digital economy is that of personal information. Many of the largest and most valuable Internet companies such as Google and Facebook are based on the business model of harvesting user data in order to accurately facilitate the targeting of advertisements, and the influencing of behaviour. This business model is one which also applies to the market for smartphone and tablet applications. At present the privacy policies of these services generally operate on a take it or leave it basis; where users either reject the gathering of their personal information by not using a service or application, or they use the service under terms and conditions which they have little control over. With the inevitable move towards increased use of cloud computing these trends can be expected to continue and even to be exacerbated as more user data will be stored at remote locations.

The OPENi project [1] provides a platform which will alterthe dynamics of user control over their personal data. Its main components are the API framework and the Personal Cloudlet framework. The API framework allows for frictionless interoperabilitybetween cloud based services, and the Personal Cloudlet framework. The Personal Cloudlet is a virtual space that securely stores user data and gives users primary control over that data. OPENi allows for users to decide which aspects of their personal data they are prepared to share by giving them the option of fine grained authorisation and access control. Therefore OPENi as a platform solves some of the problems raised, by empowering the user to take control of their digital identity and describes the user controlled privacy capabilities of OPENi.

The design and implementation of the privacy preservingand user-centric features of the Personal Cloudlet Framework is the primary focus of this paper. In section II we will outline the key objectives and other considerations that influenced their design. In section III we define the Personal Cloudlet influenced research questions. In section IV we analyse the key security technologies and similar platforms in this field. In section V we will describe how the Cloudlet Framework's privacy aware and user-centric features were implemented and how they were influenced by the objectives set out in the OPENi project.

II.BACKGROUND

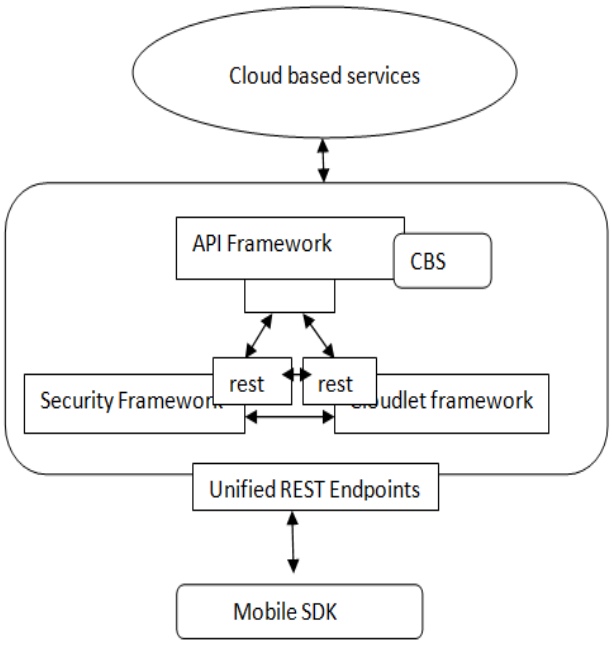

Fig. OPENi Architecture

DOI 10.17148/IJARCCE.2015.412135

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

The OPENi platform is composed of four distinct but interrelated frameworks, they are:

1) Security framework: This framework is responsible for security on the OPENi platform. The frameworks access control functionality; tightly coupled with the Cloudlet Framework; allows users to have more control over their personal data and their interactions with cloud-based services.
2) API framework: It is an open framework which operates with a number of cloud-based services, abstracting the challenges to a single open standard without affecting any service features.
3) Personal Cloudlet framework: The OPENi Personal Cloudlet framework provides users with one location to store and control their personal data.
4) Mobile Client Library: In order to provide access to the security, API, and Personal Cloudlet frameworks, OPENi architecture provides a number of mobile client libraries. One is a cross platform Javascript/HTML library for use in Apache Cordoba mobile web-apps and HTML5 and another is a native Android client library.

The combination of the API, Security, and Personal Cloudlet frameworks concept makes OPENi a very powerful and beneficial platform for consumers, application developers, and service providers.

OPENi Personal Cloudlet Objectives

1) To build a web based security and authorisation framework that will satisfy a context broker and the service provider's requirements that will enable the sharing of context information between applications in accordance with the users privacy settings and provide more control to user over their data.
2) To deliver an open source platform that will allow application consumers to create, manage and deploy their data in the cloud (Personal Cloudlet). Each Personal Cloudlet will have a credentials that will be linked to its user's identity over the web in a similar way that a social profile does today.
3) To provide and promote a novel, user-centric application experience of cloud-based services not only across different devices and platforms but also inherently across different applications. The OPENi framework will enable users to share and distribute their data across their applications.
4) To ensure the OPENi platform provides support and maintains a low barrier to entry for application developers and service providers.

III. OPENi CLOUDLET RESEARCH QUESTIONS

By taking into account the overall objectives we have designed the key research question for the OPENi Personal Cloudlet Framework as follows:

Along with providing the developer with the ease of accessing the data in a privacy concerning manner, how to a secure and provide privacy to a web based framework so as to develop it in order to provide user-centric management to late-bound and dynamic data and APIs?

In case to identify the overall research question we need to specifically negotiate the various facets: User-centric, privacy, security, scalability, late-binding, identity management, web, external applications and minimal exposure. The specific research questions that identify these facets are as follows:

1) Which mechanisms should the OPENi platform include to provide fine-grain access control in a scalable manner?
2) Which mechanisms should be included by the OPENi platform to provide its users with a braod picture of access to their data?
3) How should major and strong differences in data representation by the 3rd party applications be investigated in order to facilitate data re-use and interoperability?
4) How should the Cloudlet Framework present data to enable convenient meta-processing; both indexing and searching; to facilitate the user in monetising their data in a privacy preserving way?

How the project objectives, research questions, and other considerations relate to and influenced the platform's implementation is discussed in section V.

IV. STATE OF THE ART

OPENi deliverable D2.3 not only gives a privacy and security results identifying specific security and privacy surfaces of the OPENi solution as a whole but also provides its components: the Cloudlet Platform and the API framework. A brief discussion of this result is out of the scope of this paper and it is suggested that the reader refers to the public deliverable for such a discussion.

A. Available Frameworks

We have provided the list of existing results of Seventh Research Framework Programme projects regarding privacy and security as well as other available open solutions.

Seventh Framework Programme projects include some goals of OPENi which are relevant to privacy and security as:

PrimeLife: PrimeLife project which ran from 2008 to 2011 was funded by EU. The main moto of PrimeLife project was to "gain stable privacy and identity management to future networks and services". From many of the results of Prime Life, one of specific which was relevant to OPENi was the development of the Prime Life Policy Language (PPL) together with the design of a PPL policy engine which enables a scenario of privacy policy negotiations where the resulting policy is stored with the Personally Identifiable Information (PII), known as a 'Sticky Policy'.

A4Cloud: The A4Cloud project (known as the Cloud Accountability Project) defines the accountability policy structure for cloud services with the aim of motivating end users to "hold each provider accountable for how it manages, uses, and passes on data and other related information". The project is developing tools for

enforcement of these policies and is extending the PPL engine design from Prime Life to achieve this goal.

TCLOUDS: The main aim of TCLOUDS is not only to develop an advanced cloud infrastructure to deliver computing and storage that can achieve a new level of security, privacy and ability to recover from ouatge but also ensure it is cost-efficient, simple, and scalable.

TRESCCA: The TRESCCA project's mission is to form the base of secured and trustworthy cloud platform by making sure it has strong logical and physical security on the edge devices, by making use of both hardware security and virtualization techniques by taking into consideration the whole cloud architecture. The project provides a proposal and demonstration of hardware / software solutions which allow stakeholders to administor the processing of their sensitive data to a remote processing engine opening up whole new field of cloud services and applications

CUMULUS: CUMULUS identifies the limitations with the development of an integrated structure of models, processes and tools supporting the certification of security properties of infrastructure (IaaS), platform (PaaS) and software application layer (SaaS) services in cloud. CUMULUS structure brings all togeher including service users, service providers and cloud suppliers to work unitedly with certification authorities in order to ensure security certificate validity in the drastic changing cloud environment.

PICOS: The main aim of this project is to provide state-of-the-art platform which is open and will be able to provide trust, privacy and identity management in mobile communities.

SWIFT: Not only it identifies the usability and privacy concerns but also focuses on extending identity functions and federation to the network .

Irrespective of EU funded research, other organizations and institutions have also distributed relevant open frameworks on privacy and security. Among them includes:

- **Apache Cocoon**: is a easily adaptable module for authentication, authorization and user management.
- **Apache Shiro**: powerful and easy-to-use Java security framework that performs authentication, authorization, cryptography, and session management.
- **Spring Security**: powerful and highly customizable authentication and access-control framework.

V. PERSONAL CLOUDLET FRAMEWORK IMPLEMENTATION

In this section we describe the implementation of Personal Cloudlet Framework, the high level architecture, the software stack, and the key features. Particular emphasis is given on the features that are core to the platform it's user-centric and privacy aware nature.

A. Framework Architecture

The Personal Cloudlet Framework was built as a REST enabled web service. The microservices paradigm was used as the basics in the design and implementation of this distributed application ; a messaging framework is used for communication between the number of discrete software modules of which it is composed. A micro-services architecture is chosen because of its support for: functional separation, heterogeneity, resilience, scalability, load balancing and economy.

B. Software Stack

The following stack is chosen for the Personal Cloudlet Framework: ZeroMQ as the messaging library, JavaScript and Node.js as the programming language and runtime, Couchbase as the datastore, Mongrel2 as the webserver, and Docker as the cloud deployment platform. The NoSQLDatastoreCouchbase was selected because of the requirement for web scalable platform. Mongrel2 and ZeroMQ framework are selected for distributed applications and we selected Node.js as our preferred backend language for its native support for our data format of choice, JSON. For further details on our discussion to implement the Personal Cloudlet Framework as a distributed applicatios and technologies that we adopted to realise the application in this way please refer to OPENi deliverable D3.5 Cloudlet Platform Design.

C. User-centric & Privacy-preserving Features

This section outlines the key privacy preserving and user empowering features of the Personal Cloudlet Framework. The user attains more control over their data when compared to traditional data platforms by the combination of these features.

1) OAuth& JWT: JWT are digitally signed base64 encoded JSON objects. Sessions and claim are managed by the stateless REST based frameworks which uses JSON Web Tokens (JWT). The Personal Cloudlet Framework extendes the use of JWTs to manage the context of the party accessing the users data. In OPENi the same endpoints are used by multiple 3rd parties to request data but are presented with different subsets of user data dependent on user permissions. This data masking is enabled by JWTs in accordance with an OAuth 2.0 compliant workflow. A client key and secret key is generated when app developers register an app on the framework. Initialization of the OPENi client side library requires both these keys. When the user has to log in these keys are passed to the OAuth views on the auth server. A context is given to all other app interaction using the combination of user login and app keys. The login dialog and the permissions dialogs are the two primary OAuth dialogs. In the case of the login view the system generates a JWT with the user identifier and the app identifier embedded in it if the user credentials are valid and the apps client and secret key are valid. This JWT is then passed to the client side library where it is embedded as a HTTP header in all other interaction with the Personal Cloudlet Framework. For the framework to maintain statelessness and play a role in attaining webscalability JWTs are the key technology whereas the security is increased by the OAuth dialogs which takes user/permissions interaction off of the 3rd party

application and into an environment where the Personal Cloudlet Framework has control. Data reuse, fine-grained access control, and user control mechanisms of the Personal Cloudlet Framework are built upon the foundations set by the OAuth and JWTs.

2) Data Reusability; App Interoperability:All Personal Cloudlet data is persisted to a NoSQL Document store as outlined in the Software Stack section; accordingly a users Personal Cloudlet is composed of a number of JSON Objects. We implemented the OPENi Types as the system is designed to allow a user to share their data across multiple applications and services. The structure of the JSON objects is exposed by OPENi Types within a Personal Cloudlet without revealing the objects content ; they can be compared to an SQL schema in that it defines strict parameters to decide which data structures are validated before they are persisted to a Personal Cloudlet. Before an Object and its members are constructed they have to adhere to the strict criteria described by OPENi Types. These criteria mainly include the members: name, primitive type (int, boolean, string, data, etc), whether the datapoint is required, and an optional enumeration of accepted values. OPENi Types are allowed to inherit other OPENi Types within them to support advanced use-cases. All OPENi Types are public on the Personal Cloudlet Framework and app developers are free to create their own or reuse OPENi Types created by others. The platform provides an OPENi Type builder GUI which simplifies OPENi Type design for convenience. As the aim of the Personal Cloudlet Framework is to encourage OPENi Type reuse the platform provides a Registry which describes existing OPENi Types also gives an indication of aOPENi Type's popularity. OPENi Types are indexed and searchable on the Registry both by the number of Objects of that type and of the amount of Cloudlets using that OPENi Type. However, which Personal Cloudlets have objects of a particular OPENi Type is not indicated to preserve privacy. As OPENi Types belong to the end users as a collective app developers cannot prevent another service provider or app developer from reusing types that they created on the system. Data reuse and interoperability between devices, applications, and service providers is enabled by OPENi Types and the Registry is designed to encourage the reuse by making it easy to discover existing data structures and by their structure and by their popularity.

One of the primary research objectives of the OPENi project is addressed by the OPENi Types on the Personal Cloudlet Framework; the objective to allow the user share data, both contextual and personal, across many applications and devices (RQ3). It also oversees another of OPENi's objectives, to have a low barrier to entry for developers, as the mechanism allows for data structure reuse it makes it easier for app developers to create their apps.

3) Fine Grained Access Control:Data access is controlled by the Personal Cloudlet Framework by attaching permissions objects to each individual Object within a Personal Cloudlet. Each permissions object lists approved apps and whther the approved app can create, read, update, and delete (CRUD) that object. These permissions rules are checked before the API call is acted 41 upon when an app calls uses the objects API. The developer must include their permissions requests in a manifest which is embedded within their app similar to which the developer keys and the Object Types are embedded in the mobile application. The client side library send the permissions manifest to a server-side OAuth page when a user opens an OPENi enabled app for the first time. Along with the permission requests the user is then given the option to agree or disagree. The scope of access can be modified by the user in some cases i.e. reduce access from any data within their Personal Cloudlet to data created by the app. Users are persisted to the framework through the Permission API with both the users and the apps developers context when they agree to the app permissions. The actors are verified by the system with the help of the context and persist the CRUD updates accordingly. The Personal Cloudlet owner can edit their permissions and update access to individual objects through the User portal at any time. OPENi uses an eventual consistency model when it comes to propagating the permissions across multiple Personal Cloudlet Objects.

A separate worker checks that all Personal Cloudlet Objects have their CRUD entries updated, while the client side is informed immediately that the permissions have persisted. The permissions mechanism is one of the Personal Cloudlet Framework's key user-centric features, it addresses the research question 1 (RQ1) of how to provide a fine-grained access control in a scalable manner. The user is given full control over third party access to their data by storing the permissions with the data in a NoSQLdatastore and implementing an OAuth permissions mechanism. The user is restricted access to some of their data but is allowed access to others. It gives the user the power to address the privacy issues that arise from the cross-app data sharing that the OPENi Types encourage, as permissions are set by the user on a per-application basis.

4) User Dashboard:The user dashboard is a HTML application that gives the user access to their Personal Cloudlet. It allows them to browse all the data, view details of the apps that requested access to their data, and it also allows them to view or edit app permissions. The main features of the User Dashboard are as follows:

- Data browsing - the Personal Cloudlet owner or we say the end user can view all their data grouped by type, app, or by time. It also allows them in viewing the data in their Personal Cloudlet filtered by the permissions settings for each of their 3rd party apps.
- Auditing- Personal Cloudlets access request logs can be viewed by the user, the logs are grouped by requests allowed and requests denied. A table format or the Piwik analytics engine can be used to view the logs. We reversed the traditional paradigm of analytics engines while integrating Piwik, historically analytics engines present information regarding users interaction to the service provider, with OPENi we do the opposite. The

end user is presented the analytics on app developer and service providers interaction with their Personal Cloudlet.

- Permissions- all permissions associated with their app can be viewed and edited by the user.
- Privacy feedback –this mechanism analyses a user's Personal Cloudlet and permissions settings and presents simplified feedback and guidelines regarding their data leakage.
- Notifications– notifications are setup, attached to the user's Personal Cloudlet so that they are informed of access requests and permission requests on their Personal Cloudlet. In the OPENi client library a link to the user dashboard is embedded, however for security reasons it opens on a separate browser window where the user has to log in. The dashboard establishes the OPENi objective of giving users more control over their data and directly addresses research question 2 (RQ2). This is achieved by informing end users of activity on their Personal Cloudlet; it also affords them the ability to alter permissions if they see activity that they don't agree with.

5) Privacy Preserving Data Aggregator:The privacy preserving data aggregation (DA) component provides 3rd party services which integrate with the Personal Cloudlet Framework enables to view aggregated user data from across multiple cloudlets while concealing the cloudlet owner's identity. For each service the Personal Cloudlet owner has to explicitly opt-in to the data aggregator. It negotiates with the permissions component to identify Cloudlets that wish to share data with the 3rd party when the Personal Cloudlet Framework gets a request for aggregated data. It then collects the data from each Personal Cloudlet, aggregates it, analyses it for privacy leakage and then sends the results to the 3rd party. The purpose of the DA is enabling the e-commerce business models that depend heavily on analytics to migrate to the Personal Cloudlet Framework; to still get insight from collective data but in a more open and privacy preserving way. By providing a mechanism for convenient indexing of data this feature directly addresses research question 4 (RQ4) so that Personal Cloudlet owners can contribute to the data aggregates in a privacy-preserving way.

VI. CONCLUSION AND NEXT STEPS

This paper has described how the OPENi project implementsfine grained access control to empower users withcontrol over how applications can use their data. This is achieved through the use of REST based endpoints, objectbasedaccess control, OPENi Types, and stateless JSON Web Tokens (JWT).

The implementation allows users to share, reuse, and control access to their data across many mobile applications while maintaining cloud scalability. The open source OPENi implementation is being leveraged by a number of commercialisation projects, funded by Enterprise Ireland, for development of scalable cloud based mobile applications enriched with the privacy preserving access control mechanisms.

Work currently under way with the OPENi project includes the validation of the implementation. This validation work will be reported in follow up publications.

REFERENCES

[1] "OPENi - Open-Source, Web-Based, Framework for Integrating Applications with Cloud-based Services and Personal Cloudlets."http://www.openi-ict.eu/, accessed: 2015-01-15.
[2] "OPENi open source project," https://github.com/OPENi-ict/, accessed: 2015-01-15.
[3] A. Iosif*et al.*, "A Community-based, Graph API Framework to Integrate and Orchestrate Cloud-Based Services," in *Proceedings of AICCSA*, IEEE Computer Society, 2014, awaiting publication.
[4] M. Petychakis*et al.*, "Enterprise Collaboration Framework for Managing,Advancing and Unifying the Functionality of Multiple Cloud-Based Services with the Help of a Graph API."*Collaborative Systems forSmart Networked Environments. Springer Berlin Heidelberg*, pp. 153–160, 2014.
[5] K. Doyle and D. McCarthy, "OPENi White Paper:An End Users Perspective: Digital Identity Putting the Genie Back in the Bottle," http://www.openi-ict.eu/wpcontent/uploads/2014/07/openi whitepaper.pdf, Sept 2014, accessed:2015-01-15.
[6] D. McCarthy *et al.*, "OPENi Deliverable D3.5: OPENi CloudletFramework Design Document," http://www.openi-ict.eu/wpcontent/ uploads/2014/10/OPENi D3.5.pdf, Sept 2014, accessed:2015-01-15.
[7] R. Kleinfeld*et al.*, "OPENi Deliverable D3.6: OPENiSecurity and Privacy Specification," http://www.openi-ict.eu/wpcontent/ uploads/2014/10/OPENi D3.6.pdf, Sept 2014, accessed:2015-01-15.
[8] "Apache Cordova," http://cordova.apache.org/, accessed: 2015-01-15.
[9] R. Illera, S. Ortega, and M. Petychakis, "OPENi DeliverableD2.3: Security and Privacy Considerations for Cloudbased Services and Cloudlets," http://www.openi-ict.eu/wpcontent/uploads/2013/11/OPENi D2.3.pdf, Jan 2013, accessed:2015-01-15.
[10] "PrimeLife - Bringing sustainable privacy and identity management tofuture networks and services," http://primelife.ercim.eu/, accessed: 2015-01-15.
[11] C. A. Ardagna, L. Bussard, S. D. C. Di, G. Neven, S. Paraboschi,E. Pedrini, S. Preiss, D. Raggett, P. Samarati, S. Trabelsi, and M. Verdicchio,"Primelife policy language," http://www.w3.org/2009/policyws/papers/Trabelisi.pdf, 2009, accessed: 2015-02-25.
[12] "A4Cloud, The Cloud Accountability Project," http://www.a4cloud.eu/,accessed: 2015-02-25.
[13] "TCLOUDS: Trustworthy Clouds Privacy and Resilience for InternetscaleCritical Infrastructure," http://www.tclouds-project.eu/, accessed:2015-01-15.
[14] "TRESSCA TRustworthy Embedded systems for Secure Cloud ComputingApplications," http://www.trescca.eu/, accessed: 2015-01-15.
[15] "CUMULUS: Certification infrastrUcture for MUlti-Layer cloUd Services,"http://www.cumulus-project.eu/, accessed: 2015-01-15.
[16] "PICOS - Privacy and Identity Management for Community Services,"http://www.picos-project.eu/, accessed: 2015-01-15.
[17] "SWIFT - Secure Widespread Identities for Federated Telecommunications,"http://www.ist-swift.org/, accessed: 2015-01-15.
[18] "Apache Cocoon," http://cocoon.apache.org/, accessed: 2015-01-15.
[19] "Apache Shiro," http://shiro.apache.org/, accessed: 2015-01-15.
[20] "Spring Security," http://static.springsource.org/spring-security/site/, accessed:2015-01-15.
[21] "ZeroMQ," http://zeromq.org/, accessed: 2015-01-15.
[22] "Couchbase Server," http://www.couchbase.com/, accessed: 2015-01-15.
[23] "Mongrel2," http://mongrel2.org/, accessed: 2015-01-15.
[24] "Docker: Build, Ship and Run Any App, Anywhere,"https://www.docker.com/, accessed: 2015-01-15.
[25] "JSON Web Token (JWT)," http://jwt.io/, accessed: 2015-01-15.

DOI 10.17148/IJARCCE.2015.412135