Improving Mozilla's In-App Payment Platform

Ewa Janczukowicz[1,2], Ahmed Bouabdallah[2], Arnaud Braud[1], Gaël Fromentoux[1], and Jean-Marie Bonnin[2]

[1] Orange Labs, Lannion, France
{ewa.janczukowicz,arnaud.braud,gael.fromentoux}@orange.com
[2] Institut Mines-Telecom / Telecom Bretagne, Université européenne de Bretagne, Cesson sévigné, France
{ahmed.bouabdallah,jm.bonnin}@telecom-bretagne.eu

Abstract. Nowadays, an in-app payment mechanism is offered in most existing mobile payment solutions. However, current solutions are not flexible and impose certain restrictions: users are limited to predefined payment options and merchants need to adapt their payment mechanisms to each payment provider they use. Ideally mobile payments should be as flexible as possible to be able to target various markets together with users' spending habits. Mozilla wants to promote an open approach in mobile payments by offering a flexible, easily accessible solution. This solution is analyzed, its shortcomings and possible improvements are discussed leading to an original proposal.

1 Introduction

Smartphones have changed the way mobile payments work. Marketplaces with applications have become an essential element of the mobile payment ecosystem. They have changed users' spending patterns and got especially specialized in micro-payments [1]. There are multiple application stores that offer in-app payment functionalities, like Google's or Apple's solutions. However they are mostly wall-gardened, so clients and developers need to have an account set up with the imposed payment provider. The system is easier to control since there are no unauthorized third parties, but at the same time it becomes very limited.

PaySwarm and Mozilla have chosen a more open approach, in order to implement platforms based on open standards and accessible to multiple payment providers. So far some limiting implementation choices are imposed, but these projects are still under development. Mozilla's idea of a payment platform seems to be the most open and flexible. This approach is beneficial for new and emerging markets, since different payment methods can be introduced.

This paper focuses on Mozilla's payment solution. Firstly, it is presented and analyzed. Secondly, its limits and possible improvements are discussed. Finally a solution is proposed and analyzed.

L. Corral et al. (Eds.): OSS 2014, IFIP AICT 427, pp. 103–106, 2014.

2 Mozilla's In-App Payment Platform

In-app payments are supported by Mozilla, that encourages providing a possibility of previewing an app or installing its basic version for free [2]. It also gives the possibility of implementing different marketplaces and working with different payment providers, thus it would be possible to target various markets and to address the needs of all users no matter the payment method [3].

Mozpay is a payment solution implemented in Firefox OS v.1.0 [4, 5]. Mozilla offered a WebPayment API that, via the mozpay function, allows web content to perform a payment [6]. Figure 1 shows the existing call flow.

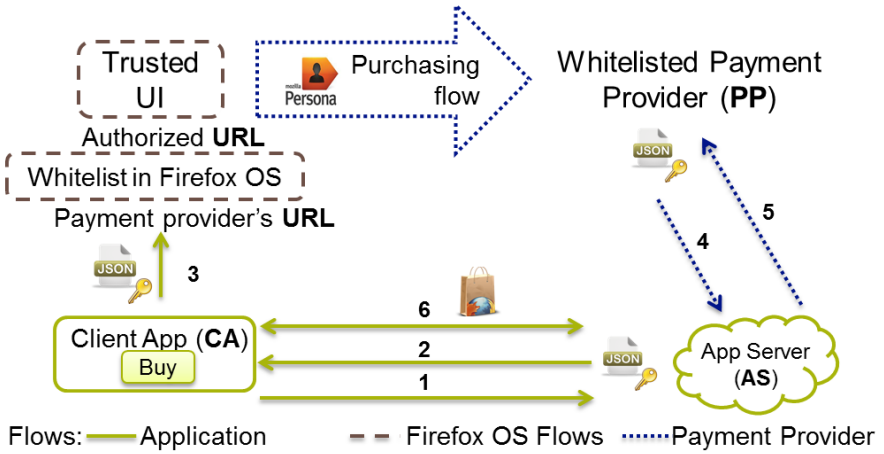

Fig. 1. Mozpay based call flow

PP implements the WebPaymentProvider API [7]. The payment flow is managed from **PP**'s server inside the trusted user interface (UI), limited to whitelisted domains. The whitelist is preregistered in the user agent and is controlled by Mozilla or whoever builds the OS. So far there is only one **PP**, created by Mozilla [2].

AS contains all application logics, manages the payment token and assures delivering goods to the user. It is assumed that to set up payments, a developer is already registered within **PP** (e.g. Firefox Marketplace Developer Hub), they have exchanged information like: financial details, application key and secret.

CA allows buying digital goods as one of its features.

The **payment token** contains all the essential information concerning a good being purchased. It is sent between all three parties throughout the payment process.

The purchase flow given below is based on Mozilla's payment provider example.

1. A user by clicking the "Buy" button requests a payment token from **AS**.
2. **AS** generates and signs the payment token, later sends it to **CA**.
3. **CA** forwards the token by calling the mozpay API [3]. If **PP** is whitelisted a trusted UI is opened, the user authenticates and the purchasing flow starts.

4. Postback (success) or chargeback (error) are tokens with additional fields (e.g. transaction ID) that inform **AS** about the payment result.
5. When **AS** receives a postback (or a chargeback) it acknowledges it.
6. In case of successful payment the purchased good is sent to the user.

The mozpay function has been proposed to be abandoned due to too rigid end-to-end transaction flows by imposing the payment token mechanism. Exposing payment provider primitives was suggested as an improvement. Payment providers would manage their own payment flows by providing JavaScript files to developers and by using a trusted UI with access restricted to whitelisted domains [8]. It can be seen on Figure 2 where the calls 1 and 2 from Figure 1 are replaced by a JS file.

3 Proposed Solution

In existing solution in order to access a trusted UI payment providers had to be whitelisted by Mozilla. It was impossible to add a new one between the Firefox OS versions. The improvement of exposing payment primitives does not solve this issue, since the access to trusted UI remains restricted to whitelisted domains [8].

A certification mechanism is a possible improvement that could replace a predefined whitelist. The solution is presented in Figure 2.

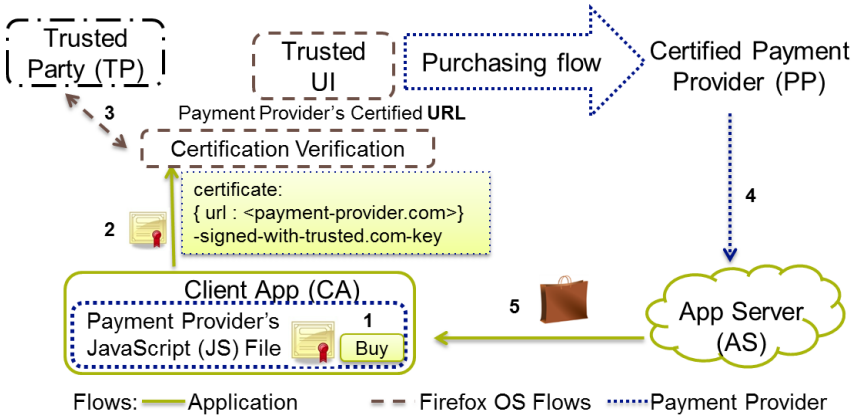

Fig. 2. Proposed solution

Instead of the mozpay method, **PP**'s JS file is included within **CA**. **PP** also provides a certificate with a URL needed to launch the trusted UI. When the Firefox OS receives a request to start the payment, it calls **TP** and verifies the certificate. If the verification is successful it uses the provided URL to open the trusted UI and the payment process begins. We assume there are no attacks on application integrity.

The proposed architecture allows changing the list of authorized payment providers without the need of redistributing the whole operating system every time a new player enters the business value chain. Instead of a central entity that controls the whitelist,

there may be several trusted parties. As a result the number of payments providers would increase, so they would compete and transaction fees would become more beneficial. This also gives clients and app developers more freedom to choose a payment option. More universal system would give the possibility of efficiently targeting specific markets and clients' spending habits. There are a lot of factors to consider: different type of clients (illiterate, cash-challenged, without credit cards) and national regulations (taxes, currency). Additionally, well-known, open standards would facilitate the development process.

The drawback of the solution is that payment providers would need to adapt their flows in order to assure certificate management. Security aspects need to be studied, although an advantage of certificates is that they are widely implemented and trusted.

4 Conclusion

In-app payments are used more often but the widely used application stores or payment providers have implemented a walled-garden approach. Mozilla wants to change the way in-app payments work by offering a platform that is open and that targets new markets while not imposing strict business models. The version implemented so far has several limits. One of the biggest limitations is a whitelist of authorized payment providers that is currently shipped with the devices. The proposed solution solves this problem by offering a certification system that would manage payment providers. As a result Mozilla's solution can become more flexible and be able to meet most of participating players' requirements.

References

[1] Copeland, R.: Telco App Stores – friend or foe? In: IEEE 14th International Conference on Intelligence in Next Generation Networks (ICIN 2010), Berlin, Germany, October 11-14 (2010)
[2] https://developer.mozilla.org/docs/Mozilla/Marketplace/Marketplace_Payments (accessed November 27, 2013)
[3] https://hacks.mozilla.org/2013/04/introducing-navigator-mozpay-for-web-payments/ (accessed November 28, 2013)
[4] https://developer.mozilla.org/en-US/Firefox_OS (accessed November 29, 2013)
[5] Janczukowicz, E.: Firefox OS Overview. Telecom Bretagne Research Report RR-2013-04-RSM (November 2013)
[6] https://wiki.mozilla.org/WebAPI/WebPayment (accessed November 28, 2013)
[7] https://wiki.mozilla.org/WebAPI/WebPaymentProvider (accessed November 28, 2013)
[8] https://groups.google.com/forum/#!msg/mozilla.dev.webapi/cyk8Nz4I-f4/5er6JojC3TsJ (accessed November 13, 2013)