A Middleware for Securing Mobile Mashups

F. Batard, K. Boudaoud, M. Riveill
I3S Laboratory- CNRS/University of Nice Sophia Antipolis
B.P 143 – 06903 Sophia Antipolis
{batard,karima, riveill}@polytech.unice.fr

ABSTRACT
Mashups on traditional desktop devices are a well-known source of security risks. In this paper, we examine how these risks translate to mobile mashups and identify new risks caused by mobile-specific characteristics such as access to device features or offline operation. We describe the design of SCCM, a platform independent approach to handle the various mobile mashup security risks in a consistent and systematic manner. Evaluating an SCCM implementation for Android, we find that SCCM successfully protects against common attacks such as inserting a malicious widget from the outside.

Categories and Subject Descriptors
D.2.m [**Software Engineering**]: Miscellaneous – *reusable software*. D.4.6 [**Operating Systems**]: Security and Protection – *access controls*. H.3.3 [**Information Storage and Retrieval**]:Information Search and Retrieval – *Information filtering*. H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web-based services*.

General Terms
Design, Security.

Keywords
Mobile mashups, mobile widgets, secure communications.

1. MASHUP SECURITY ISSUES
Security of mashups has been the subject of several works [1][2][3][4][5]. The security risks introduced by mashups (i.e. widgets aggregation) are mainly due to the vulnerabilities of widgets that represent the weakest link. Thus, securing this latter implies mainly preventing malicious widgets to be added in a mashup and restraining them to communicate with other widgets in a same mashup or with the outside.

In the current context where mashups try to convince the public of their usage and easiness, mashup providers are quite reluctant to restrain developers' creativity as they rely on them to provide as many widgets as possible. With the emergence of the application store concept providing trusted applications for mobiles, this concept may rise also in mashup context. A mashup would have then to provide a secure library of trusted widgets and possibly use technical solutions to ensure their integrity. One of these solutions can be the Widget Digital Signature proposed by W3C [6]. As any signature mechanism, it provides a clear way to prove the origin of a widget and its author. However, this technology is often considered difficult since mashups do not provide simplified way to sign their content, as it is the case for example with Firefox plug-ins. Therefore, it is important to define a simple mechanism, even if based on Widget Digital Signature, to allow secure adding of widgets.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

Allowing users to mix proprietary widgets with public ones within the same mashup can generate potential security vulnerabilities. Actually, due to browsers processing, a poorly architected mashup environment might allow malicious widgets to gain access to data held in other widgets or, to push and pull malicious server data. In fact, the host servers used for mashups can be fooled: a request coming from a corrupted widget can appear as coming from an end-user browser. Thus, what should be done is to define policies to restrain malicious widgets communications by blocking malicious exchanges between widgets and potentially isolate a malicious widget from the rest of the mashup.

Considering the security problems related to mashups and focussing on mobile context, we will now see the resulting security requirements for our middleware called SCCM (Secure Communicative Context Moshup).

2. SECURITY REQUIREMENTS
Basing on the security issues discussed previously, we have identified several security requirements that our middleware will have to fulfil. Here are some of these requirements:

- **Widgets origin and integrity checks**: SCCM needs a certain trust in the widgets added in order to prevent phishing.
- **Protection of inter-widgets interaction**: SCCM should be protected against Frame Phishing. Each widget should be totally independent from another one and should not have direct access to other widget's code.
- **Internal resources access**: SCCM has to restrict abusive or unauthorized access of the widgets to the phone services.
- **External resources access**: If widgets access resources on the web, SCCM needs to have strict regulation of the information they send and receive.
- **Moshup communication link**: SCCM should be confidential enough to prevent other softwares present on the mobile to spy on the internal moshup exchanges.
- **Protection against XSS & CRSF**: SSCM should handle these attacks.

3. Global view of the SCCM architecture
Figure 1 gives a global view of the SCCM architecture we propose and shows the security mechanisms used in the different components. As shown in this figure, our architecture is composed of five kinds of components:

- **Widgets**: are pieces of standalone web codes using HTML/JavaScript to perform actions, discuss with each other and with the phone API. If needed, they can request resources from the web. They need to be signed and this signature is checked at their loading.
- **Web Engine**: interprets the widgets code written in web languages. It enables widgets sandboxing, manages errors and uses JavaScript framework for widgets communications.

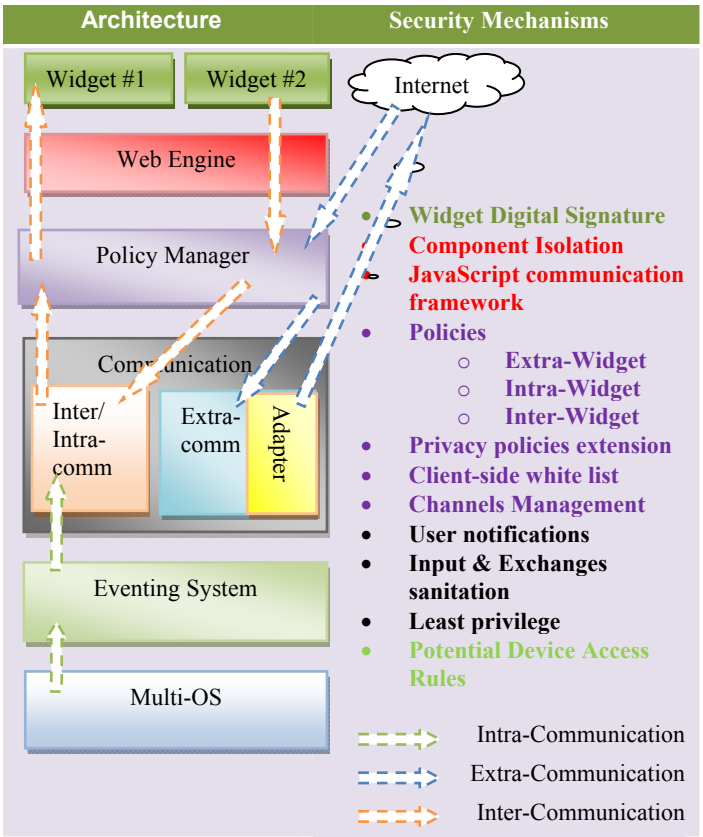

Figure 1: SCCM architecture and security embedded

- **Policy Manager**: is in charge of access regulation and authorization of the whole system. It controls who can talk to whom and potentially the content of the messages exchanged. It controls access over the channels on which the widgets and *eventing system* publish their messages. In extreme cases, the *policy manager* might request the user's decision to grant or restrict accesses.
- **Communication system**: relays information to the right recipient using two components:
  - **Intra/Inter-communication** component that relays information locally. It can transmit to a widget a message coming from another widget or from the *eventing system*.
  - **Extra-communication** component that manages the exchanges between the widget and the Internet. This component has an element called *adapter* to transform web services responses to events understandable by the widgets requesting information from Internet.
- **Eventing System**: is in charge of relaying events from the OS to the *communication system*. More specifically, it transmits, through the *inter/intra-communication* component, the data provided by the device resources (e.g. Geolocation, short messaging, etc.) to the widgets requesting that data.

For the implementation of SCCM we have chosen the Android platform for its ease of development and its potential growth on the market.

4. Security evaluation

We have tested the robustness of SCCM against ten kinds of attacks that aims to corrupt SCCM by trying to: 1) insert malicious widgets or alter valid widgets from the outside and 2) usurp widget or OS identity, retrieve private information or interrupt SCCM. The different attack attempts have shown that SCCM is robust against all attempted attacks except DoS (Denial of Service) and attacks against privacy. In this paper, we present two kinds of attacks.

Attack 1: The goal of the first attack was to add a malicious widget into a moshup. We considered that the malicious widget is already on the user's phone as it could have been downloaded on purpose or due to a phishing attack. The attacker tries to add the malicious widget in the moshup by placing it in the widgets folder. At every startup of SCCM, when the widgets are loaded, SCCM verifies the signature file embedded in widgets packages. As the widget will not have any signature file, SCCM will not allow it to be added in the moshup and therefore will not be processed.

Attack 2: The next attack targets widgets that do not have restricted topic and on which a malicious widget can subscribe and publish messages. The best attack in such situation is to use XSS and CRSF attacks to allow a regular widget to perform bad actions and give sensitive information. As the *inter/intra-communication* component provides a set of regular expressions to prevent XSS attacks and script insertions, this kind of attacks could not succeed.

5. Conclusion

In this paper, we highlighted the risks inherent to mashups and the challenges of introducing this approach on mobiles devices. Then, we have proposed a middleware called SCCM that allows three kinds of communications (inter, intra and extra) and secures these communications. In its first version, SCCM has been implemented on an Android platform. To evaluate our solution, we have conducted several security tests. For future works, we plan to work on a second version of SCCM enabling SCCM to block malicious widgets from collecting data and transmitting it outside the mobile device.

6. References

[1] F. Batard, K. Boudaoud and M. Kamel. Web 2.0 Security: State of the Art. In Proc. of the 5th Conference on Network Architectures and Information Systems Security, May 18-21, 2010, Menton, France.

[2] C. Jackson and H. J. Wang. Subspace: Secure Cross-Domain Communication for Web Mashups. In Proc. of the16th International World Wide Web Conference (WWW2007), Banff, Alberta, May 8-12, 2007

[3] F. D. Keukelaere, S. Bhola, M. Steiner, S. Chari, and S. Yoshihama. SMash: secure component model for cross-domain mashups on unmodified browsers. In Proc. of the 17th International World Wide Web Conference (WWW2008), Beijing, China, April 21-25, 2008.

[4] S. Crites, F. Hsu, H. Chen. OMash: Enabling secure web mashups via Object Abstractions. In Proc. of the 15th ACM conference on Computer and Communications Security (CCS), Alexandria, USA, VA, Oct. 27- 31, 2008.

[5] A. Barth, C. Jackson, W. Li. Attacks on JavaScript Mashup Communication. In Proc. of the Web 2.0 Security and Privacy 2009 (W2SP 2009), Oakland, California, May 21, 2009.

[6] M. Cáceres, F. Hirsch, M. Priestley. Digital Signatures for Widgets. W3C Candidate Recommendation 24 June 2010. http://www.w3.org/TR/widgets-digsig/