JSXC: Adding Encrypted Chat with 3 Lines of Code

Klaus Herberth, Daniel Scharon, Matthias Fratz, and Marcel Waldvogel
Department of Computer and Information Science
University of Konstanz, Konstanz, Germany
Email: <first>.<last>@uni-konstanz.de

***Abstract*—If it isn't on the web, it doesn't exist. However, most of our current arsenal of web services are provided *for free* by large international corporations – free as in targeted advertising. More privacy-aware self-hosted alternatives frequently lack the feature set of their commercial rivals, leaving users to decide between privacy and functionality. Therefore, we present WISEchat (Web-Integrated Secure Enhanced Chat), our concept for enhancing practical security for web-based chat, as well as an implementation, the JavaScript XMPP Client (JSXC). By design, JSXC can be easily and painlessly integrated into existing webapps to equip them with encrypted chat capabilities, making them more attractive and thus more frequently providing a secure alternative as the most functional and convenient alternative.**

I. Related Work

There are plenty of web- or browser-based chat clients. The most widely used one is presumably the one integrated in the Facebook social network; however this client is firmly tied into its intended service, and provides an equivalent level of privacy – while the connection to the server is secured by TLS, it critically provides no security whatsoever against the server. In stark contrast to Facebook's automatic logging of chat messages, Off-the-Record messaging [1] provides end-to-end encryption and confidentiality with full forward security, preventing even a malicious or compromised chat server from reading or altering messages in transit. This is clearly a highly desirable feature for any chat client that is to be integrated into services offering *privacy by design*.

Crypto Cat [2] appears to be the first popular add-on that provided an in-browser OTR-encrypted chat. As an add-on, it can profit from the full security of a native application running on a properly secured host, but requires explicit installation by the user which, for many use cases, is a crucial step too much. WISEchat removes this step by integrating directly into the web application itself, with minimal changes to the host application – generally 3 to 5 lines, depending on the degree of integration. What sets it apart from other integrable clients such as Candy [3] or Jappix Mini [4] is its set of features and its focus on security. Like Converse.js [5], it can piggy-back the authentication of the host web application,[1] saving the user another annoying login, and tearing down one further barrier between the user and encrypted chat.

II. Improving Practical Security

To avoid these barriers for the user, WISEchat is designed not as another web service, but as an open-source, XMPP-based end-to-end encrypted chat client that can be easily integrated into existing web applications with minimal effort. Its integrated nature means that there is no WISEchat server or website. It is part of whatever services it is integrated with; it is, to a certain degree, invisible. This improves upon existing, standalone Off-the-Record chat applications by providing security by usability – by offering the secure alternative as the most convenient alternative, directly in the webapp the user actually wants to use at the moment, without forcing a media break.

WISEchat is a concept; JSXC [6] is an *implementation* of WISEchat, employing widely used, highly interoperable protocols – by using standard Off-the-Record encryption [7] compatible among implementations, by using the Extensible Messaging and Presence Protocol [8] over BOSH [9], or by employing SRTP [10] for encrypted video chat. For the user, this interoperability means that (s)he can chat with any other XMPP user, regardless of the other user's choice of client or XMPP server.[2] This makes JSXC a first-class citizen among OTR-capable XMPP clients; the user might use JSXC integrated into an open-source groupware suite like SOGo [11] while reading email in the webmail interface, but prefer to chat using a standalone chat client like Pidgin [12] with its OTR plugin while in the office. The user thus keeps his/her freedom of choice, and can use whichever client (s)he finds most convenient at any given time. Security profits because secure chat is simply the most convenient option in a greater number of situations.

JSXC is designed as an implementation of open protocols, not as a web service. In order to provide cross-platform functionality, it uses open standards like WebRTC and can thus provide video chat *without plugins* in most modern browsers.[3] Combined with its use of common, widespread libraries for both XMPP and OTR encryption, this further reduces the burden on both the maintainer as well as the service administrator by keeping the number of components to a minimum. Because of its loose coupling (fig. 1), it also preserves the freedom of choice for the operator of the service it is integrated with. It uses XMPP towards the chat server, which can be set up using any one of a number of popular

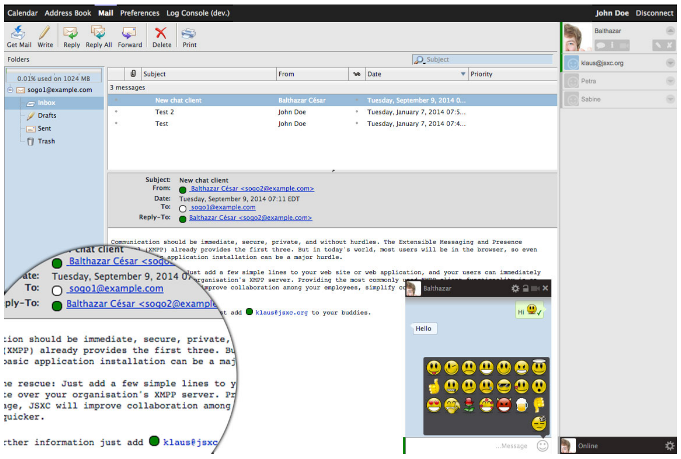
[1]This is especially helpful for webmail integration, as the user's Email address and Jabber ID are almost always identical.

[2]XMPP, being a federated protocol, will forward messages as necessary between properly configured servers, much like Email forwarded between domains. Therefore, while generally configured with a central XMPP server for each organisation, there is no need for users to connect to the same server to be able to communicate, neither with JSXC nor with any other client.

[3]As of writing, Internet Explorer does unfortunately not yet support WebRTC and thus is not supported for video chat. It can of course still be used for general, OTR-encrypted text chat.

```
<link href="lib/jquery+plugins.min.css"
     media="all" rel="stylesheet" type="text/css" />
<script src="lib/jquery+plugins.min.js"></script>
<link href="lib/jsxc.all.css"
     media="all" rel="stylesheet" type="text/css" />
<script src="lib/jsxc.all.js"></script>
<script src="js/jsxc/config.js"></script>
```

Fig. 1. Typical HTML snippet required for JSXC integration. JSXC consists of a handful of JavaScript code files, some CSS styles and several images. Apart from making these files accessible on the server and setting up a BOSH proxy for the XMPP server, the lines above are the only change required to the application itself. In some cases, it may ease maintenance to include the individual libraries separately instead of concatenating them into a small number of files. This is particularly true for the jQuery libraries, which is already present in most modern webapps for functionality other than JSXC.



	Subject	From	Date
3 messages			
	New chat client	Balthazar César	Tuesday, September 9, 2014 0...
	Test 2	John Doe	Tuesday, January 7, 2014 07:5...
	Test	John Doe	Tuesday, January 7, 2014 07:4...

Subject: New chat client
From: ● Balthazar César <sogo2@example.com>
Date: Tuesday, September 9, 2014 07:11 EDT
To: ○ sogo1@example.com
Reply-To: ● Balthazar César <sogo2@example.com>

Communication should be immediate, secure, private, and without hurdles. The Extensible Messaging and Presence ... (XMPP) already provides the first three. But in today's world, most users will be in the browser, so even ... basic application installation can be a major hurdle.

...e rescue: Just add a few simple lines to your web site or web application, and your users can immediately ...e over your organisation's XMPP server. Providing the most commonly ... ...age, JSXC will improve collaboration among your employees, simplify co... ...quicker.

...rther information just add ● klaus@jsxc.org to your buddies.

Balthazar
Hi
Hello

Fig. 2. JSXC integrated into SOGo. The user list on the right-hand side shows the online status of chat buddies. The chat window in the bottom right can be slid away when not in use, reducing to the size of its title bar. This allows the user to keep track of ongoing conversations without obliterating the screen. Also note the display of users' online status next to email addresses. This functionality is entirely provided by JSXC; SOGo does not have to support the status icons or in fact be aware of them.

XMPP server implementations – and this XMPP server itself acts as a standalone service that can be used by other desktop, mobile or web clients. In essence, this means that a single *real* service – the XMPP server – can provide any number of *perceived* services – the web chat integrated in numerous other services – thereby greatly reducing maintenance effort.

JSXC does not have to be a separate part of the web application that shares screen space but otherwise does not interact with it. For example, it can automatically detect Jabber IDs (JIDs) in the web page it is integrated on, show the online status of the user alongside, and turn them into links to start a chat with this user. This is implemented without active cooperation of the host application, keeping interdependence at a minimum without sacrificing interaction. Because JIDs are almost universally identical to a user's email address, this simple functionality already provides significantly improved usability and integration.

For example, in the SOGo integration example (fig. 2), this means that the user can instantly see whether the sender of an email is currently available for chat, and may thus be able to directly ask for clarification if need arises, saving what could be several lengthy email round trips explaining what, exactly, the recipient did not fully understand. However, because JSXC permits temporary, anonymous accounts, this feature can also turn a static documentation page into a semi-interactive medium by offering a quick, uncomplicated way of contacting the contact persons given on the page – or of seeing that they are offline and that an email is indeed necessary.

It does not have to be limited to such simple integration however. For example, a federated social network application such as buddycloud [13] could very easily provide a way of contacting users and showing their status wherever they are mentioned, without explicitly displaying the user's JID. It might then complement this by providing an easy path from the chat window to the remote user's profile, or provide some form of notification of new content posted by the remote user in the chat window. While such applications currently fall firmly into the realm of future developments, and may require deeper integration thus reducing JSXC's advantages of simple deployment into new or existing services, they clearly show that the WISEchat concept could be used as a building block for innovative web-based interaction.

III. Demonstration

We will demonstrate an example of a JSXC integration into an existing webapp. From the users' point of view, we will demonstrate the login, unencrypted chat and starting of an encrypted chat, while showing the messages as seen by our "honest but curious" XMPP server. We will also show a brief video chat conversation, but then proceed to take a look under the hood and show what changes have been made to the webapp, including the JSXC configuration and installation of the BOSH proxy.

References

[1] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-record communication, or, why not to use PGP," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. ACM, 2004, pp. 77–84.

[2] Crypto cat. Accessed Nov. 2014. [Online]. Available: https://crypto.cat/

[3] Candy chat. Accessed Nov. 2014. [Online]. Available: http://candy-chat.github.io/candy/

[4] Jappix mini. Accessed Nov. 2014. [Online]. Available: https://mini.jappix.com/

[5] Converse.js – a free and open-source XMPP chat client for your website. Accessed Nov. 2014. [Online]. Available: https://conversejs.org/

[6] K. Herberth, M. Waldvogel, and D. Scharon. (2014) JavaScript XMPP client. [Online]. Available: http://www.jsxc.org/

[7] I. Goldberg and the OTR Development Team. (2014) Off-the-record messaging. [Online]. Available: https://otr.cypherpunks.ca/

[8] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120 (Proposed Standard), Internet Engineering Task Force, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6120.txt

[9] I. Paterson, D. Smith, P. Saint-Andre, J. Moffitt, L. Stout, and W. Tilanus. (2014) XEP-0124: Bidirectional-streams over Synchronous HTTP (BOSH). Draft Standard. [Online]. Available: http://xmpp.org/extensions/xep-0124.html

[10] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," RFC 3711 (Proposed Standard), Internet Engineering Task Force, 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3711.txt

[11] Inverse, Inc. (2014) Sogo: Open source groupware. [Online]. Available: http://www.sogo.nu/

[12] Pidgin, the universal chat client. Accessed Nov. 2014. [Online]. Available: https://www.pidgin.im/

[13] Buddycloud. Accessed Nov. 2014. [Online]. Available: http://buddycloud.com/