

SECURITY IN TWITTER CLIENTS

INTECO-CERT

The present document meets the accessibility standards of the PDF format (Portable Document Format).

This document is tagged and structured, with alternatives to every non-textual element, language marking and adequate reading order.

For further information on the production of accessible PDF documents consult the guide available in the section [Accessibility > Training > Manuals and Guides](#) of our website <http://www.inteco.es>.

INDEX

1.	INTRODUCTION	4
1.1.	Motivation and Objectives	4
1.2.	Security In Twitter Clients	5
1.2.1.	Authentication	5
1.2.2.	Encryption	7
1.2.3.	Password storage	8
1.3.	Methodology	8
2.	TESTING OF CLIENTS	10
3.	STATISTICS	13
4.	CONCLUSION AND RECOMMENDATIONS	17
5.	REFERENCES	18

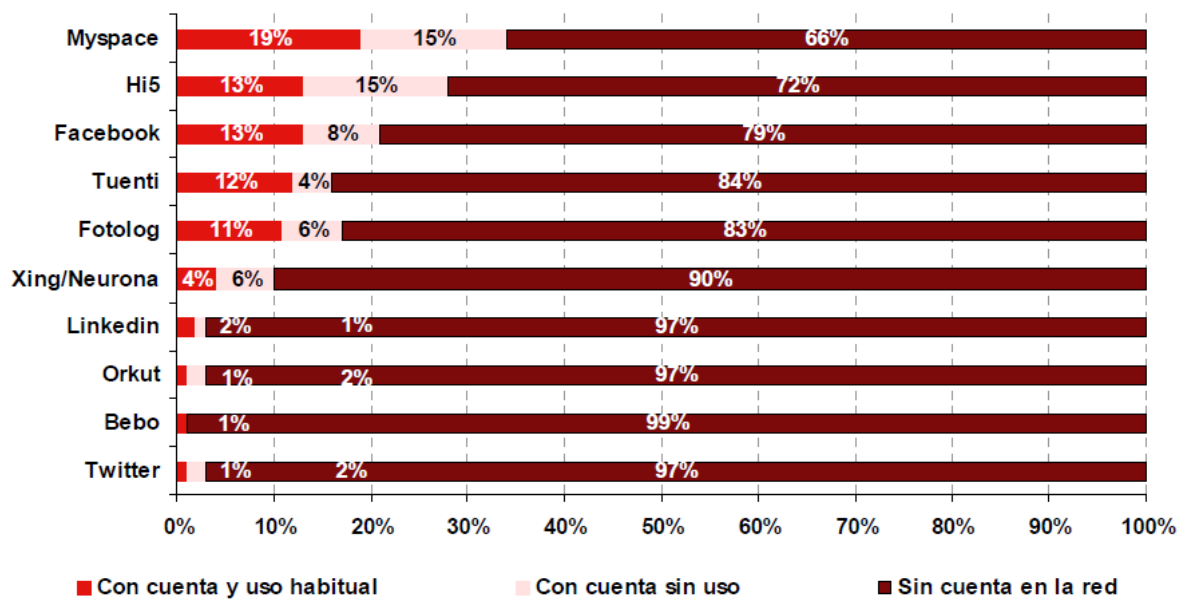
1. INTRODUCTION

1.1. MOTIVATION AND OBJECTIVES

Over the recent years, one of the pillars of the so-called Web 2.0 has been the user-generated content, which was initially present in the shape of blogs, comments and information in social networks.

We can take “microblogging” as the confluence of two of these types of information dissemination. On one hand, it is a blog simplification, reducing the size of its entries to the minimum information unit: a thought, an image, a link, a question, etc.; but, on the other hand, it may also be considered as the evolution and independization of status messages from instant messaging and social networks.

Diagram 1: Penetration of Social Networks in Spain (July 2008)



Source: INTECO, from the Observatory of social network evolution (The cocktail analysis)

Although there are many microblogging services (Tumblr, Plurk, Emote.in, Squeelr, Jaiku, identi.ca, etc.), the most well-known is Twitter¹, which, providing a very simple service focused on accessibility and easiness for the user to watch and generate content, enables unprecedented ubiquity and “freshness” of information. In addition to the web interface, the

¹ Study about privacy of the private data and the information security in the social networks from INTECO [1].

company itself offered from the start access through SMS in order to send and receive updates, thus making the service accessible to people who have no smart phone or who are not very often in front of a computer. This emphasis on mobility has also caused the user-handled updates to occur more continually, from any place and at any time.

Virtually from the beginning, Twitter made available an interface of access to its services (API – *application programming interface*) for third-party developers, contributing to the creation of clients for any platform, as well as parallel services of searching, tracking, statistics, etc.

The easiness to access the Twitter API, together with the functional simplicity of a client of this service, has enabled the emergence of countless unofficial applications to access it. According to the statistics of use [4], 78% of users gain access to the platform through applications distinct from the official web interface.

The objective of this study is the revision of the security implemented in a great variety of these clients, thus examining those aspects regarding authentication and client-server communication encryption, as well as password storage, in order to verify if the rapid growth of the platform has put security aside, entailing great risk of identity theft to the users of these clients.

1.2. SECURITY IN TWITTER CLIENTS

The security considerations for microblogging networks are usually focused on the very information spread across them, but this study is based on a more technological feature, the communication between the clients used by users and the dissemination host.

Twitter currently facilitates a series of security recommendations for the users of its API, but this was not so at the beginning and, consequently, many clients have some weaknesses in this respect.

In the following sections we describe the different aspects relating security of clients, potential threats and Twitter recommendations.

1.2.1. Authentication

Just like in any application in which there are user profiles, it is necessary to identify oneself with a user name and verify that you are the owner of that account. This security mechanism is applied both to the official web interface and to the Twitter API available for application developers.

Firstly we will revise the process of authentication on the official website, where you can find a form to introduce your account number and password. After sending them and once checked that these are correct in the server, this returns a token, a random string which univocally identifies our session, the well-known *cookie*. The browser stores it and sends it in subsequent requests in order to not have to send the credentials in each one of them.

It is important to understand that after the initial verification of credentials, the only thing that identifies a session authenticated with the server (*a priori*) is the cookie. Therefore, if someone obtains that string and makes requests including it, the server will interpret that they come from the legitimate user. This is what is called session hijacking, which permits to phish a user while the cookie is valid and the session is active (as for web applications it may be days, weeks ...)

There are mechanisms to prevent this type of attacks by checking additional parameters of the requests (origin IP, for instance) but these are not included in the object of this survey.

As regards the API, even though the communication is carried out through *http*, forms and cookies are not used as authentication mechanisms. Twitter offers two different methods of authentication, as detailed in the following sections.

1.2.1.1. Basic authentication

This is the authentication method chosen for the API in its beginnings. User credentials are sent in clear text in the headers of the *http* request. The absence of any type of encryption allows anyone who can see the request to be able to find out the used password.

1.2.1.2. OAuth

OAuth is an open protocol which permits the authorization of secure access to an API in a simple and standard way for web and desktop applications.

The process can be summarized in the following steps:

1. The user asks a desktop/web client application to access his/her account.
2. The application requests Twitter a random token which will be associated to the user-application pair.
3. The application redirects the user to the official website with the random token as parameter.
4. The user authenticates him or herself and confirms the permission for the client application to access his/her account.
5. If it is a desktop client, Twitter provides a security PIN. As for a web application, it is directly redirected to the client application, already with access.
6. The user introduces the security PIN in the client application, which therefore becomes authorized to access his/her account.

From July 2009, Twitter offers the possibility of using this authorization mechanism for applications which want to access its API. This is currently the recommended method, since it provides a greater level of security than the basic authentication for two reasons: the

credentials do not travel in each request and, above all, it is not necessary to entrust them to third-party applications.

Twitter demands the applications wanting to make use of OAuth to access its API to be registered in its web page (http://twitter.com/oauth_clients)

1.2.2. Encryption

Communications in publicly accessible networks such as the Internet *a priori* must not be considered as confidential or private, since the information travels through network resources which are not controlled by the user.

The interception and listening of third-party transmissions may be implemented, for instance, in both wired and wireless local networks with relatively advanced techniques, but which may be easily applied by means of specific tools available to anyone (network sniffers). This allows obtaining, among other things, access credentials to secured systems, with the resulting threat of identity theft.

In order to prevent it, the communication between the user and the application must be encrypted by using some cryptographic algorithm that provides confidentiality, together with an adequate protocol to manage the encryption.

The most-known standard protocol in web communications is probably the SSL/TLS on 443 port. Generally, you can know if it is being used when the accessed URL begins with *https*.

Twitter web does not use encryption by default, since when typing the URL address www.twitter.com in the browser, it accesses <http://twitter.com>, instead of redirecting to <https://twitter.com> (which is also available). Nevertheless, when after filling the form with the user name and password we make the authentication request, this does redirect you to a secure URL (<https://twitter.com/sessions>), the credentials thus travelling encrypted through the network.

However, again in subsequent requests, the cookie travels in clear text across the network, enabling the above mentioned session hijacking attacks.

The situation is much more delicate if a basic authentication against the API is used, since the credentials travel in each request and capturing one of those packages would be enough to find out the user password, which may be used to phish user identity at any time the attacker considers appropriate.

As for OAuth authentication the problem is similar to web authentication. The credentials never travel through the network between the third-party applications and the Twitter servers (the encrypted authentication system of the official website is directly used), but like the cookie, the authorized token is indeed sent in each request. If a secure encryption protocol is not used, the same session hijacking attacks may occur.

1.2.3. Password storage

As it was pointed out previously, the clients that do not use OAuth protocol for authentication/authorization must know the user name and password to access a profile. If we do not want to introduce them in every connection to the server (whether to send a tweet or to receive updates), it is necessary for the applications to store those credentials.

Regardless of the location in which they are saved (configuration file, log, database...), the authentication data can either be saved in clear text or encrypted with a reversible algorithm to prevent anyone else who might access those data from reading them without the required key.

When using encryption there are different possibilities of implementation. Some clients opt to implement a self-made encryption method, although be it with standard algorithms, using a fixed key. A generally more secure mechanism is to use the password management functionalities provided by environments such as KDE (Kwallet), GNOME (GPass) or AIR (encrypted local store)

There is some controversy about which storage mechanism is more advisable. Obviously, the well-implemented encryption provides a greater degree of security, but there are those who state that the security of permissions at file system (or database) level should be sufficient (at least in modern operating systems) and that, in practice, the majority of implementations do not offer increased security [3].

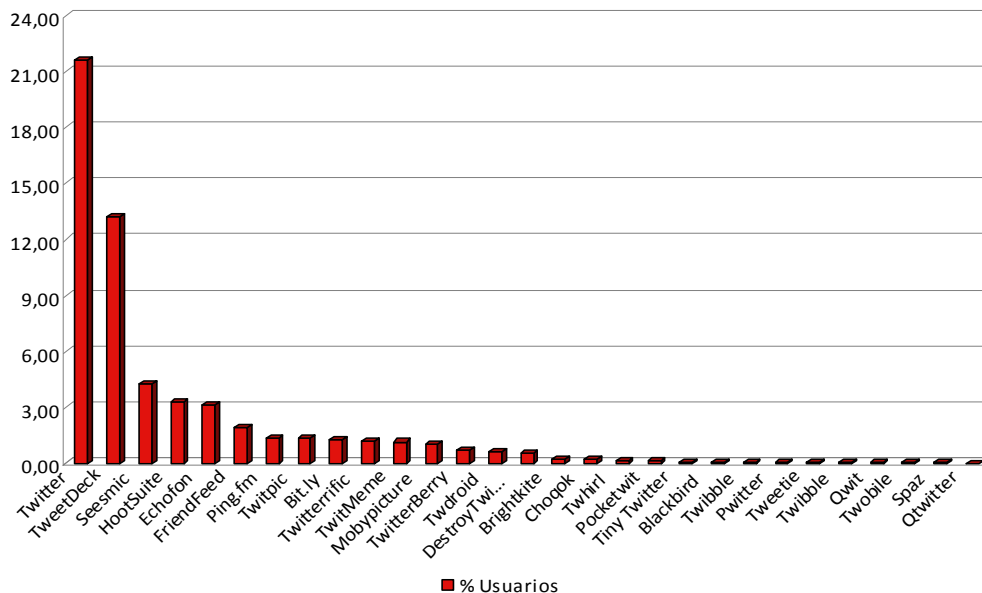
In any case, it is unquestionable that the fact of giving the passwords to a third-party application and having them stored in an additional location (apart from the Twitter servers) is a less secure option than when the user simply memorizes it and provides it only when necessary.

1.3. METHODOLOGY

When choosing the clients to use, we have employed the statistics of global use of clients available in twistat.com [4], 03 November 2009 (Diagram 2). Even though it is not an official source, the statistics are taken from the Twitter API itself, which carries out a tracking of the clients with which each tweet is sent, so they should be considered as trustworthy.

In view of the impossibility to find statistics of use per platform, we have firstly opted to take the more widely used clients. In order to cover the whole range of considered platforms (Windows, Linux, MacOS, iPhone, Android, Blackberry, WindowsMobile), some of the most popular clients available for them have been included, even though their use does not reflect a significant percentage out of the total.

Diagram 2: Statistics of use of Twitter clients (11/03/2009)



Source: twittstat.com [4]

For each client we have carried out tests to determine their security characteristics:

- **Authentication mechanism:** It has been checked for each client whether the mechanism used to access the API is basic authentication or OAuth protocol. Due to the differences in the process, it is immediate to distinguish between them by simply adding an account to the clients
- **Encryption of communications:** The information on the use of SSL in the clients has been obtained by recording the network traffic generated by the clients, verifying that in those in which the encryption is optional, the option actually accomplishes its function. As for web clients, it is impossible to know how the communication with the Twitter API is carried out. In this special case it is taken into account whether a secure protocol for the user access to the application is employed.
- **Password storage:** For those clients that use basic authentication and, therefore, need to store the user password, it has been attempted to specify whether its storage is implemented in plain text or with encryption. This information has been mainly taken from manual tests or, failing that, from the client's documentation and, in some cases, from the source code itself.

When it comes to multi-platform applications, most of them use a portable framework or language, e.g. AIR, .NET, Java or Python. In these cases, assuming that the security characteristics are similar in any system, the most convenient option to perform the testing was selected.

2. TESTING OF CLIENTS

The results of the tests performed with the selected Twitter clients are summarized next.

The meaning of each column, although self-explanatory enough, is detailed below:

- **Version:** The version of the application used to carry out the testing or, failing that, the date in the case of web platforms, in view of the absence of any accessible version number.
- **Percentage of users:** Percentage of users using the client, according to the statistics taken as reference (see section 1.3 on *Methodology*)
- **Platform:** In this column the platform used to test the client is specified, regardless of whether the application is available for additional systems or not.
- **Authentication:** Authentication protocol used to access the API, Basic or through OAuth, as specified in section 1.2.1.
- **SSL:** It determines whether encrypted connections are employed or not, or whether it is an option to activate.
 - Desktop/Mobile clients: If the client uses encrypted communications to access the API.
 - Web clients: If the communication between the user and the web platform uses encrypted connections (*HTTPS*)
- **Password encryption:** It specifies whether the password storage in the device executed by the client is encrypted or not. This is not applicable in web platforms due to the impossibility of knowing the infrastructure used in their servers.

Table 1. Results of testing with Twitter clients.

Client	Version	% of Users	Platform	Authentication	SSL	Password encryption
Twitter	11/03/2009	21,6	WEB	N/A	Optional	N/A
TweetDeck (Desktop)	v0,31,4	13,23	Windows	Basic	Yes	Yes
Seesmic (Desktop)	v0,66	4,22	Windows	Basic	Yes	Yes
HootSuite	11/03/2009	3,32	WEB	Basic	No	N/A
Echofon	v2.1.1	3,11	Iphone	Basic	Yes	Yes
FriendFeed	11/03/2009	1,9	WEB	Basic	Yes	N/A
Ping.fm	11/03/2009	1,35	WEB	Basic	Optional	N/A
Twitpic	11/03/2009	1,35	WEB	Basic	Optional	N/A
Bit.ly	11/03/2009	1,3	WEB	Basic	No	N/A
Twitterrific (MacOS)	v3.2.1	1,2	Mac	Basic	Yes	
Twitterrific (iPhone)	v2.1	1,2	Iphone	Basic	Yes	Yes
TwitMeme	11/03/2009	1,15	WEB	OAuth	No	N/A
Mobypicture	11/03/2009	1	WEB	Basic	No	N/A
Mobypicture (Android)	v20091910	1	Android	Basic	Yes	Yes
TwitterBerry	v0.9.1.1	0,7	Blackberry	Basic	Yes	Yes
Twdroid	v2.7.1	0,65	Android	Basic	Yes	Yes
DestroyTwitter	v1.7.2 Beta	0,58	Windows	Basic	Yes	Yes
Brightkite	v1.2.2	0,25	Android	Basic	Yes	Yes
Choqok		0,2	Linux	Basic	Optional	Yes

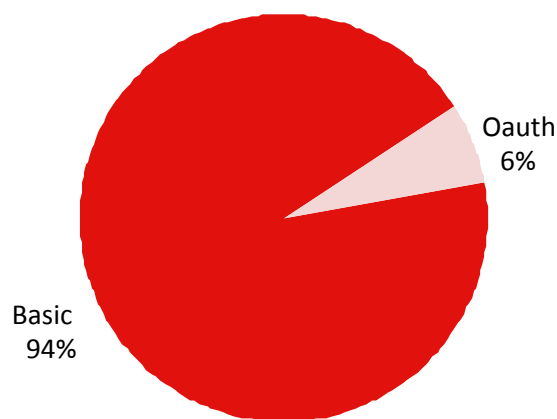
Client	Version	% of Users	Platform	Authentication	SSL	Password encryption
Twirl	v.0..9.4	0,17	Linux	Basic	Yes	Yes
Pocketwit	v75b	0.14	Wmobile	Basic	No	Yes
Tiny Twitter	v1.8.4	0.1	Wmobile	Basic	No	No
Blackbird	v0.5.23	0.05	Blackberry	Basic	Yes	
Twibble	V0.8.3	0.05	Blackberry	Basic	Yes	Yes
Pwitter	V1.1.6	0.05	Mac	Basic	Yes	Yes
Tweetie	v1.2.4	0.05	Mac	Basic	Yes	Yes
Tweetie	v2	0.05	Iphone	Basic	Yes	Yes
Twibble	v0.5.6	0.05	Linux	Basic	Yes	Yes
Qwit	v1.0-Beta	0.05	Linux	Basic	Optional	Yes
Twobile	v1.7.5	0.05	Wmobile	Basic	Optional	Yes
Spaz	v0.8.2	0.03	Linux	Basic	Optional	Yes
Qtwitter	V0.6.6	N/A	Linux	OAuth	Yes	N/A
Twitter	11/03/2009	21.6	WEB	N/A	Optional	N/A
TweetDeck (Desktop)	v0.31.4	13.23	Windows	Basic	Yes	Yes
Seismic (Desktop)	v0.66	4.22	Windows	Basic	Yes	Yes
HootSuite	11/03/2009	3.32	WEB	Basic	No	N/A

Source: INTECO and twittstat.com (% of use)

3. STATISTICS

Different diagrams reflecting the distribution of clients for each of the analysed security features are shown next.

Diagram 3: Authentication Mechanisms per percentage of clients

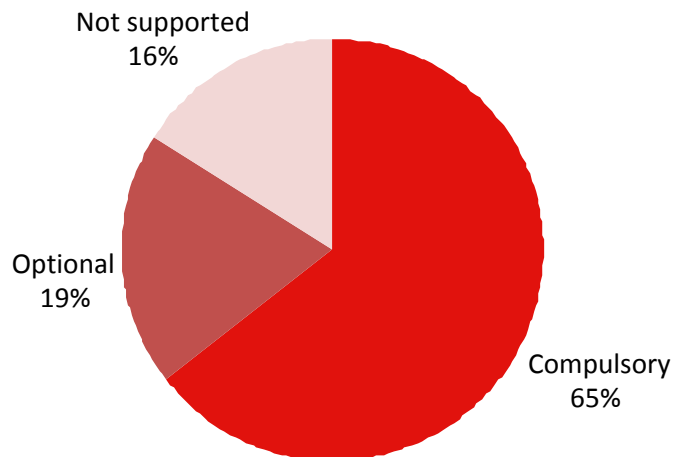


Source: INTECO

As regards the authentication mechanism, Diagram 3 shows that the use of OAuth in comparison with the basic authentication is minimal. This may have two reasons:

- The basic authentication process was the only one available originally and OAuth has only been available since July 2009. It is difficult to persuade developers of previous applications to change something that is working properly.
- Though developers know it, some argue that the initial process of user account authorization is too cumbersome [5].

Diagram 4: Use of encryption in communications

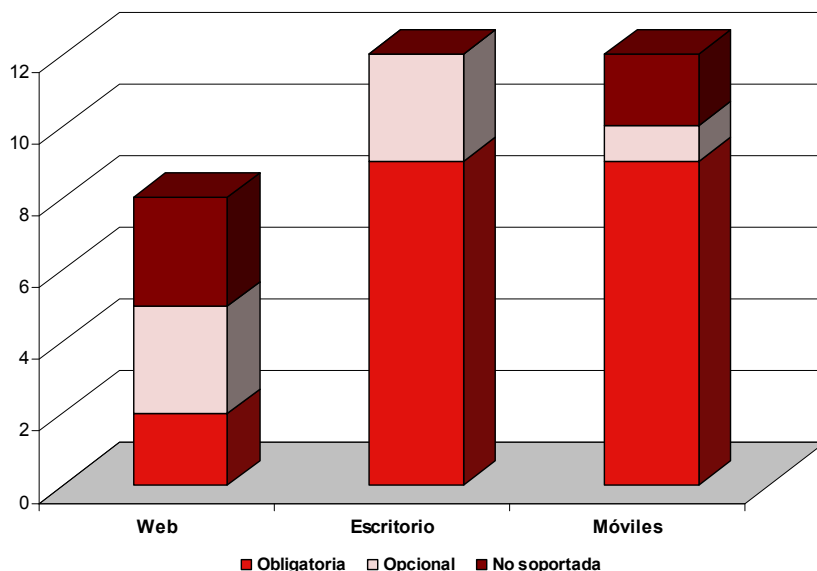


Source: INTECO

As for the encryption in communications (Diagram 4) its use is widely spread (84%), although it is activated by default in a rather lower percentage (64%).

It is important to emphasize that OAuth authentication is used in only one of the cases in which the encryption is not supported. This means that in the rest of the cases the theft of credentials would be immediate with the capture of just one user request.

Diagram 5: Use of encryption in platform communications



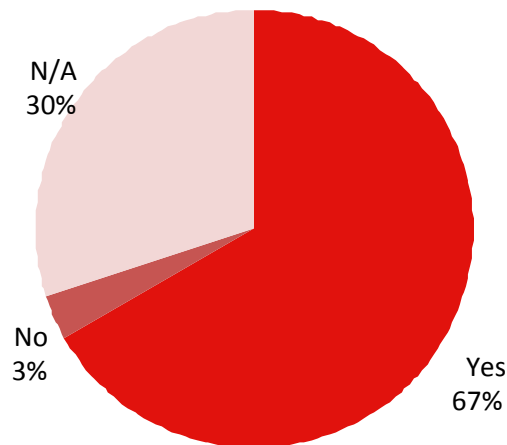
Source: INTECO

As it is shown on Diagram 5, the encryption is much more used in different desktop and mobile platforms. It is surprising that, in a platform where the secure connection mechanisms like *http/ssl* are so widely spread for other applications, these are not used more often. We have to remember that for web applications these data refer to the client-user communication, and it would be desirable to be able to check how the communication between the application and the Twitter API is carried out, in order to ensure that the communication route is confidential in all its stages; therefore, the situation could be even worse than what the statistics show.

It is remarkable that, among the mobile platforms, the clients not supporting encryption are all for Windows Mobile, which has only one client with optional encryption. In the rest of mobile operating systems it is compulsory.

In Linux systems the encryption is mostly optional, which may be seen in accordance with its possibilities of customization and with a profile of advanced user who prefers to control the configuration of his/her applications. In every Mac and Windows clients, the use of secure communications is compulsory.

Diagram 6: Encrypted storage of passwords



Source: INTECO

When the applications need to store the user credentials because of using basic authentication, we can see that nearly all those in which it could be checked apply the recommended practices. Nonetheless, it would be necessary to verify whether the used encryption mechanisms actually provide a sufficient degree of confidentiality to guarantee the security of our passwords.

In most cases standard password storage mechanisms are used, which we can consider secure; however, it has been confirmed that a fixed key with an easily reversible XOR encryption is used at least in one of the applications (Qwit).

4. CONCLUSION AND RECOMMENDATIONS

After analysing the security features of the most used Twitter clients the following conclusions can be drawn:

- The use of OAuth is not spread, but it is to be expected that it will be adopted sooner or later, given the intention of Twitter to eliminate the support for the basic authentication [2]. This mechanism has significant advantages as regards security, and we believe that in spite of its seemingly greater complexity, once its use becomes popular and there are functional libraries for the majority of languages, all the clients will eventually migrate to this solution, for the benefit of users.
- Even though the basic authentication is used, most clients make use of encrypted connections in a way that the user is protected against theft of credentials and session hijacking.
- Nevertheless, although there is protection against third-party attacks, the access credentials are still left to applications of which the level of security or the legitimacy cannot be determined.

Taking these points into account, from INTECO we suggest the following recommendations:

- The users must check the security characteristics of the clients, looking for a compromise between functionality and desired security level, and giving preference to those clients using OAuth and encrypting communications.
- Application developers must know and follow, as far as possible, the recommendations of Twitter [2], which may be summarized in using OAuth for authentication and encrypting communications for any access requiring authorization.
- As for Twitter, there are possible measures to enhance security: to offer users to make use of SSL for every request as an option in the account configuration (just like Google does with Gmail), in a way that if it is activated, it automatically redirects your requests to the HTTPS site.

5. REFERENCES

1. [Study on personal data privacy and information security in online social networks](#)
INTECO, 2009.
2. <http://apiwiki.twitter.com/Security-Best-Practices>
3. [http://developer.pidgin.im/wiki/PlainTextPasswords.](http://developer.pidgin.im/wiki/PlainTextPasswords)
4. [http://www.twitstat.com/twitterclientusers.html.](http://www.twitstat.com/twitterclientusers.html)
5. <http://blog.atebits.com/2009/02/fixing-oauth/>