Newcastle University

School of Computing Science

CSC8299 Individual Project

MSc Computer Security and Resilience

Safety first - maintaining privacy and trust in whistleblowing services

Lorenzo Primiterra

Supervised by Dr. Budi Arief

August 2013

CSC8299 Individual Project: Safety first - maintaining privacy and trust in whistleblowing services

L. Primiterra
School of Computing Science, Newcastle University
L.Primiterra@ncl.ac.uk

Supervisor: Budi Arief

Abstract. Whistleblowing is a phenomenon that has always existed, but recently it has taken on a renewed importance within society. Unfortunately, our society has always labelled whistleblowers as spies or evil doers, and the law has done little to support their cases. Blowing the whistle can be a risky action as whistleblowers can lose their jobs or being persecuted. Many software tools have been developed to give them the ability to retain their anonymity while bringing to light the nefarious activity. This paper examines many open source whistleblowing platforms and attempts to analyse them in terms of their variable features. This project aims to improve the GlobaLeaks platform by developing a mobile frontend called iGlobaLeaks, which can be used by any whistleblower to report leaks anonymously from any sector or region. The entire development process will be described from the early mock up designs to the final implementation. Moreover, this project has also contributed to the documentation effort of the GlobaLeaks platform API for future developers.

Declaration: I declare that this dissertation represents my own work except where otherwise stated.

1 Introduction

Recently, the activity by the United States government to censor the work of Edward Snowden has brought *whistleblowing* to the forefront of most media outlets. This term refers to the practice of publishing wrongdoings at work and consequently bringing them to the attention of one's employers or a relevant organisation.

Whistleblowers are defined as "current or former organisation members or persons whose actions are under the control of the organisation, who lack authority to prevent or stop the organisation's wrongdoing" [*1*]. They have always been victimised, associated with sneaks, spies, squealers and other despised forms of informer [*2*].

A whistleblower is often considered untrustworthy. Transparency International research (2009) shows that "Across the 10 EU countries, the term 'whistleblower' is associated with being an informant (e.g. in the Czech Republic, Ireland, Romania and Slovakia), a traitor or spy (Bulgaria, Italy) and/or a snitch (Estonia, Hungary, Latvia and Lithuania)" [*3*] [*4*]. And in some countries, like Ireland or Italy, there is a mistrust

of public authorities and a diffused behaviour on not to speak out against your neighbour or colleague.

Whistleblowing can be viewed as a process. The whistleblower can be at some time a member of the organization to which wrongdoing is ascribed and usually lacks the authority to prevent or stop the organization's wrongdoing. Because of this, anonymity is a crucial factor in this process; "anonymous whistleblowers may be practicing caution and discretion but informants do not, sometimes can not, always respect the anonymity of their sources" [***2***].

Challenging or accusing a higher authority always involves risk. Because of this, whistleblowers need courage and conviction, since their actions involve an ethical conflict between personal and organisation values (Jubb, 1999).

A questionnaire survey of whistleblowers who contacted Whistleblowers Australia [***4***] shows that 34 of the 35 subjects had been victimised as a result of exposing the misconduct. Some relevant data from the results include breakups of their significant relationships in which they directly blamed the whistleblowing. 29 subjects experienced symptoms they attributed to stress, 13 subjects were forced by their employer to see a psychiatrist and 2 had attempted suicide.

What is emerging from this research is how the (not anonymous) whistleblowing dramatically affected their life. However despite what happened, 23 subjects said that they would do it again because as they state, "Deep down I know I did the right thing, and by my doing, it may help others to do the right thing" [***4***].

Brian Martin [***5***] describes the typical recommendations for whistleblowers:

- Collect large amounts of documentation about the problem including records of meetings and email.
- Behave in an exemplary way and document it; it is easy for rumours to spread without control, and this is often followed by derogatory comments.
- Do not trust the official channels; when sending the report to your boss you cannot ignore the fact that he might be involved directly.
- Be prepared for reprisals. Reprisals include ostracism, harassment, threats, spreading of rumours, reprimands, referral to psychiatrists, demotions, forced transfers, dismissal and blacklisting.

The most recent story of a big whistleblowing case appeared on every newspaper and television in the Western world involving Edward Snowden, the former CIA technical worker who leaked a lot of sensitive information and wrongdoing about US agencies. In particular Snowden revealed a system called Prism, a surveillance system launched in 2007 by the US National Security Agency that is able to receive emails, video clips, photos, voice and video calls, social networking details, logins and other data held by a range of US internet firms [***6***]. With this system, US agencies gathered millions of phone records and monitored Internet data, according to Snowden [***7***].

Snowden, who has now been granted a temporary asylum in Russia [***8***], "is the world's most famous spy, whistleblower and fugitive, responsible for the biggest intelligence breach in recent US history" [***9***].

Interesting is MacAskill's question about Snowden in his article on The Guardian "So is he whistleblower or traitor?" [***9***]. The debate is not easy, but when asked about the motivations of his whistleblowing, Snowden answered, "I don't want to live in a society that does these sort of things... I do not want to live in a world where

everything I do and say is recorded" [*7*]. It is interesting that this is the same answer presented in chapter 1, when showing the results of the Australian's whistleblowers.

1.1 Motivations

The motivations of whistleblowers are varied but tend to centre on several important aspects. Most whistleblowers usually start the complaint within the organization through what they considered were the proper channels. Near and Miceli [*1*] claim that "knowledge of extra-organizational channels would stimulate whistleblowing by signalling that society desired and would support legitimate whistleblowing".

Currently, technology offers us more tools to protect whistleblowers and guarantee a secure and private channel for them to 'blow the whistle' whilst maintaining their confidentiality [*5*], but most of them are highly expensive.

In order to address this issue, several projects have been set up. One example is 'GlobaLeaks' an open source project started in 2011 by Hermes team, Center for Transparency and Digital Human Rights in Milan Italy. It aims to support whistleblowing in many diverse scenarios by providing a tool that can be used by anyone.

The team's main goal is to build universal software, easily accessible by anyone, to simplify the task of the whistleblower and protect their privacy whilst giving all the tools to the administrator who manages the whistleblower platform.

However, this is not their only objective; they also aim to "promote transparency and adoption of more open practices as fundamental values for spotting malpractice throughout society" [*10*]. This will not be achieved by merely building a piece of software, but also by disseminating information about the general benefits of whistleblowing and keeping an eye on the updates on this theme to ensure the team is following the right path. One of the main drivers in carrying out this project is to make a real positive impact to help the whistleblowers' cause, i.e. not solely to develop software for a dissertation project that would be abandoned after the completion of the degree.

1.2 Project Goals

The project aims *to investigate the privacy and trust issues surrounding whistleblowing services and to implement a prototype in collaboration with an existing, open source whistleblowing platform.*

The intended objectives are as follows:

1) Research existing whistleblowing software solutions in terms of their strong and weak features regarding privacy, trust and security.
2) Focus the investigation on the GlobaLeaks open source software and compare the GlobaLeaks software with other whistleblowing services found in Objective 1.

3) Use the research undertaken in Objective 2 about different whistleblowing software to formulate specifications for the development of a stable and secure iPhone app.
4) Design and implement an iPhone app, according to the specification defined through Objective 3.
5) Test and evaluate the developed iPhone app with the GlobaLeaks open source community.

1.3 Project Scope

The scope of the project is defined according to the requirements and deliverable outcomes set for the project. The overview of the project report is given below as different stages of the project:

Stage 1 (Introduction): In this stage, many issues and challenges related to whistleblowing are presented and analysed. There is a brief introduction and description of the project.

Stage 2 (Technical Background): This stage shows a panoramic view of existing whistleblowing platforms and their technical details with an in-depth study on GlobaLeaks and Tor platforms.

Stage 3 (Requirements Analysis): This area describes the criteria to compare existing whistleblowing software/platforms. Furthermore, the software and platforms analysed in Stage 2 are compared using this set of criteria.

Stage 4 (Design and Implementation): In this stage, the iPhone software and plugin are designed and implemented. Before the implementation, an overview about various techniques is shown.

Stage 5 (Testing and Evaluation): Testing and Evaluation stage includes the testing of the plug-in developed for GlobaLeaks and the iPhone application with a community of possible users and an evaluation of them by the open source community of GlobaLeaks.

Stage 6 (Conclusion and Future Work): This describes how whistleblowing will be considered in the future, based on future laws and software, and how the solution developed in this project will potentially be integrated into and supported by the GlobaLeaks project.

1.4 Dissertation Structure

This dissertation is divided into 7 chapters; chapter 1 outlines the motivation and objectives of this work. In chapter 2, various whistleblowing platforms are presented and compared and contrasted on their features. Then an introduction on the GlobaLeaks project followed by more technical details about the project and short overview of Tor, which plays a fundamental role in this work are given. The process of requirement analysis is presented in chapter 3, along with a brief introduction of

developments tools on iOS platform and the reason of this choice. chapter 4 presents the design and implementation choices according to the last iOS standards and informed by the research carried out in this project. In chapter 5 the results of the testing process are summarised and in chapter 6 an evaluation of the project and a usability analysis are presented. Finally, in chapter 7 there is a conclusion and a panoramic view of the future work and implementation of GlobaLeaks.

2 Related Work

The most famous organization in the whistleblowing field is surely WikiLeaks[1], an international non-profit organization who publish leaks and secret information received from anonymous sources. WikiLeaks claimed to have a database of more than 1.2 million documents within one year from its launch and has produced more scoops in its short existence than the Washington Post in 30 years [***11***].

The 'WikiLeaks' model is a completely new concept for journalism. The first WikiLeaks model was similar to Wikipedia, where every user can edit and add information; but more recently WikiLeaks has used different models, some more open and others more narrowed, to maintain a balance between control of the leaks and delivery of information.

"From the moment WikiLeaks abandoned its first 'wiki-fied' model, it has functioned much like any other big media player, top-down editorial control with Assange as the gatekeeper" [***11***].

So should WikiLeaks be a model for future whistleblowing open projects? Definitely yes, but it will never be a substitute for good journalism. Logically, journalism should look at the WikiLeaks model and become more open, rather than WikiLeaks to become a news delivery system.

Currently, WikiLeaks is banned in the US, and the Columbia University's School of International and Public Affairs "recommended that current students not tweet or post links to WikiLeaks, because doing so could hurt their career prospects in government service" [***12***]. PayPal and Amazon blocked Assange's accounts; but this model taught everyone in the field a lesson difficult to forget, "Freedom of information and expression is a core value of our institution" [***12***].

In the next chapter, several whistleblowing open source software projects will be analysed technically to show how they achieve anonymity and security in the whistleblowing process along with their pros and cons.

There are many more closed whistleblowing platforms that will not be analysed because there is nothing much to analyse apart from their submission interface as the source code and the security implementation is private, a list can be found at [***13***].

[1] http://wikileaks.org/

2.1 Survey of existing whistleblowing platforms

2.1.1 AdLeaks

The first platform to analyse is AdLeaks, which is a research project by Freie Universitat Berlin, FX Palo Alto Laboratory and Stevens Institute of Technology. This is not a complete project, it provides a submission frontend, but it lacks the backend necessary to securely manage and distribute received disclosures. It does offer interesting design ideas for building a good whistleblowing platform.

The aim of their project is to "protect against an adversary who can see most of the traffic in a network, such as national intelligence agencies with a global reach and view" [***14***]. "The objective of AdLeaks is to make whistleblower submissions unobservable even if the adversary sees the entire network traffic" [***14***].

The way they achieve this is to hide all the information exchanged from the whistleblower to the platform and into online advertising, so basically AdLeaks is an online advertising network. The ads carry additional ciphered code, and a whistleblower browser substitutes the ciphertext with encrypted parts of a disclosure. They claim that with this protocol an adversary can't distinguish between the regular browser's transmission and the modified whistleblower's transmission. Ads are ubiquitous and the whistleblower never has to use a particular website to communicate with the platform.

This infrastructure consists of guards, aggregators and decryptors; the guards manage the encryption of the request, the aggregator aggregates the ciphertexts it receives per second and transmits them to the decryptor, finally the decryptor decrypts the downloaded ciphertexts and, if it finds data in them, reassembles the data into files [***14***].

In the fields of encryption and decryption performance, the platform is quite fast, a 12-core Mac Pro can serve 51480 whistleblower at the same time, but the performance of the upload/download link is really bad, it can take days to transfer a few megabytes. Another weak point of this approach can be that the whistleblower must use it on a personal machine, where nobody else has access, and the adversary has no control over users' end hosts and does not block Internet traffic. The other big issue with this approach is the software distribution; it cannot be offered online because the adversary would be able to observe the submission. The solution to this problem is to use an approach of "security through obscurity" in this case distributing the software with CDs, DVDs and removable media bundled with a movie or audio CD to make it invisible.

So in conclusion AdLeaks demonstrates that a similar approach is possible and flexible but very difficult to implement in a good software product.

2.1.2 Briefkasten

Briefkasten is a German secure web application that allows submitting content anonymously in a reasonably secure way. According to the authors "We do not see this project as a variation of WikiLeaks. We do think, though, that in our time,

providing our users with a reasonably secure way to transfer sensitive data to us is a simple question of editorial hygiene" [*13*].

They published the source code and some specification on their GitHub repository[2], the main feature of this platform is to not store any login credential, in order to protect the identity of submitters, instead it uses watchdog monitoring software, ideally installed in another machine or network to perform regular end to end tests of a briefkasten instance.

When the platform receives an attachment, briefkasten sanitizes it to remove a number of meta-data that could compromise the submitters identity, GPG[3] encrypt it and send it via email to a pre-configured list of recipients. Upon successful upload the submitter receives a unique URL with a token that he or she can use to access any replies the recipients may post. That reply is the only data persisted on the server.

This application does not actually provide much resistance to Communications Data Traffic Analysis [*13*], so they do recommend the use of Tor [*15*].

2.1.3 Honest Appalachia

Appalachia is a rural and isolated region in the United States and because of that, politicians and corporations too often go unobserved, so Jim Tobias, a University of Pennsylvania-trained journalist, and his co-founders decided to start this project; they were inspired by WikiLeaks but with a local focus.

Honest Appalachia is a secure website[4] where whistleblowers can anonymously leak documents to the public without fear of reprisal [*16*]. They claim not to publish a document until it is reviewed to ensure it is genuine and true, this solves the problem of the local newspaper that doesn't have enough staff or money to dig into long investigations [*17*]. "We're trying to take the risk out of whistleblowing," Tobias says, "People are scared to death of going public with anything. But they know that there's a lot going on" [*17*].

Honest Appalachia is distributed as an open source project[5] that anyone can modify, adapt and reuse, the secure upload site runs behind a Tor Hidden Service [*15*], when it receives a file, it encrypts it and uploads it to a different file server, then securely deletes the original copy. The secure copy is only accessible to members of Honest Appalachia who strips it from metadata and distributes it among their network of journalists. Their goal is to have a system with a good balance between security, ease of use, and cost effectiveness.

This project is not the most technologically advanced, but it offers a lot of ideas about how a whistleblowing platform should be advertised, how the staff behind a whistleblowing service should behave and how easy should be to use, even if the authors suggest that users not access Honest Appalachia from a work computer, because it can be monitored.

---

[2] https://github.com/ZeitOnline/briefkasten
[3] http://www.gnupg.org/
[4] https://honestappalachia.org/
[5] https://github.com/honestappalachia

2.1.4 The GlobaLeaks project

GlobaLeaks is developed by a non-profit entity (Hermes centre[6]), the first version 0.1 was released in 2011, a second version 0.2 was entirely rewritten starting from scratch in may 2012; a first alpha of this version was released in May 2013 and the beta version was released in July 2013, the stable version is expected to be released in September 2013.

In GlobaLeaks, every topic may have an appropriate whistleblowing site, called context; the roles separation is well specified, users are divided in whistleblowers, node administrator and receivers (Journalists, experts, public official).

The aim of this project is to reduce the entrance barrier to implement whistleblowing procedure; it offers an easy installation package for Linux to quickly set up a whistleblowing node. The platform runs as a Tor Hidden Service [***15***] to offer anonymity and confidentiality but can be accessed also from the normal Internet.

The GlobaLeaks platform will be analysed more in detail in chapter 2.3.

2.1.5 Comparison of Existing Whistleblowing Software

Platform	**Pros**	**Cons**
AdLeaks	- Best project in terms of security and anonymity.	- Really slow to transfer files. - It is just a prototype now, not a fully implemented software.
Briefkasten	- Implements the basic security features needed for a whistleblowing platform.	- UI not graphically advanced. - Not too much customizable.
Honest Appalachia	- Small and lightweight platform.	- Needs to be forked and improved. - Hasn't been updated in the last two years.
GlobaLeaks	- Flexibility: multiple contexts and receivers. - User friendly and easy to install. - High security level (supports Tor, GPG and HTTPS).	- Still in beta version.

This project aims to improve a whistleblowing platform and after a research and a survey of various existing platforms. The GlobaLeaks project has been selected as the base of this dissertation, since it has the support of a great team and it is still constantly updated with new features. Furthermore it is a project that has a great

[6] http://logioshermes.org/

balance between the security features and user friendliness through its clean and simple UI, it is multi platforms and has a easy and simple installation and configuration process.

2.2 The Tor Project

Tor is an acronym for The Onion Router. It is an open source software used defend a user's online personal privacy and anonymity; it prevents against any kind of traffic analysis and monitoring on the network [***15***]. "Onion Routing" refers to the distributed overlay network used to encrypt and anonymise the TCP connections as web browsing, email and instant messaging.

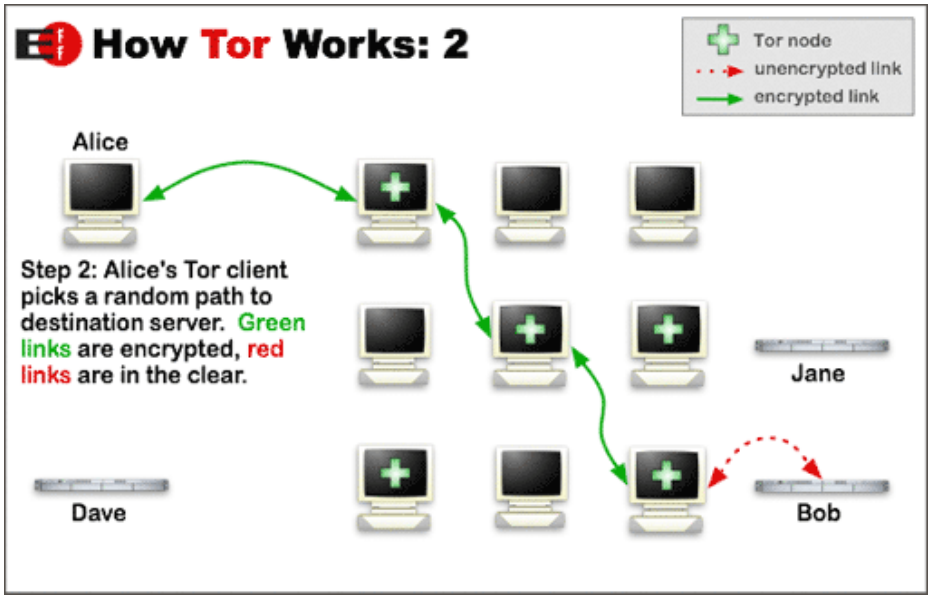

Fig. 1. How Tor works[7]

Each client chooses a path through the network and builds a circuit; each node of this circuit knows only its predecessor and successor, but no other nodes in the circuit (See Figure 1). The traffic flows encrypted in the circuit, "rather than using a single multiply encrypted data structure (an onion) to lay each circuit, Tor now uses an incremental or telescoping path-building design, where the initiator negotiates session keys with each successive hop in the circuit" [***15***]. This was necessary to prevent a hostile node from recording traffic and compromising successive nodes in the circuit.

Tor uses a generic SOCKS proxy to support almost all TCP-based software without any modification, so multiple applications can share the same circuit to improve efficiency. Tor also implements a congestion control to detect congestion or flooding in nodes and send less data until the congestion subsides and performs

[7] Credit: http://torproject.org

integrity checking to verify data integrity before it leaves the network to prevent altered data.

Tor is mainly focused on usability; it should not modify the user's everyday behaviour and does not require patching the OS kernel. It must be as transparent as possible to the user and support as many platforms as possible.

Each user that runs a local Tor software is a node of the network, it is used to fetch directories, establish circuits across the network, and handle connections from user applications [***15***]. Some of these nodes can be configured as exit nodes, the final relay that Tor traffic passes through before it reaches its destination. They advertise their presence to the entire Tor network so any users can use them, and the final destination of the connection will see the exit node as source of the connection.

All users can run a Tor exit node, according to their available bandwidth, but they should be aware that if any users do something malicious or illegal, the exit relay may take the blame. "People who run exit relays should be prepared to deal with complaints, copyright takedown notices, and the possibility that their servers may attract the attention of law enforcement agencies" [***18***].

Web server under the Tor network uses a .onion suffix and until a few years ago could only be accessed from users inside the Tor network. A new tool, tor2web, now at a stable development version, allows users to access any .onion web server hosted under the Tor network from the regular web. Cheng affirms that "the existence of tor2web will encourage more organizations to publish content anonymously through Tor, now that such a heavy access restriction has been lifted" [***19***].

This tool is very important for a GlobaLeaks web server that runs inside the Tor network, which will be explained in more details in the next chapter.

2.3 Enhancing GlobaLeaks

The GlobaLeaks project is composed of several sub-projects, each one with a specific goal to achieve. They are "necessary to ensure a high degree of privacy, distribution of responsibilities, ease of use and operations of the GlobaLeaks site [***10***].

The project is composed of:

- GlobaLeaks software, composed of Client and Backend.

The client is a modern JavaScript application to allow whistleblowers, node administrators and receivers to interact with the GlobaLeaks server.

The backend is the actual software installed by the node administrator, written in python; it is responsible to handle the whistleblower submissions and notification to the receivers. The backend is composed by six components, *the submission component* which deals with how information is acquired by the whistleblower, *the storage* responsible to deal with material uploaded by the whistleblower, *the status page* deals with the interactions, such as authentication and receipts, *the notification system* deals with availability of new submissions or updates, *the delivery systems* deals with the delivery of new submission to receiver and finally *the administration* component deals with setup of the node and its settings.

- Anonymous Python Application Framework (APAF)

This is a container to allow anybody to build an application that can be published automatically to the Tor network as Tor Hidden Service. According to the

GlobaLeaks developers "GlobaLeaks' aim is to reach even non-technically proficient users and enable them to run whistleblowing software. Having a desktop application that can be run simply by clicking on an executable drastically decreases the entrance barrier"[10]. And, because the GlobaLeaks server runs under the Tor network as Tor Hidden Service, this framework produces the desirable output.

- tor2web

As described in the chapter above, tor2web allows people to run a server in the Tor network that is accessible from the outside. This facility allows the node administrator to expose its GlobaLeaks server on the web without registering a domain. For example, a domain that in the tor network is denoted as *http://ymasaptkeux3qgbc.onion* could be visited from a normal browser with the URL *http://ymasaptkeux3qgbc.tor2web.org*

- LeakDirectory

This is an archive of all the GlobaLeaks sites and whistleblowing related material, when setting up a node the admin can choose to have it published on LeakDirectory. The aim is to have a unique node list to help whistleblowers choose which one to submit information or leaks.

- PrivacyBadge (Torcheck)

This is just a plugin to verify that users have configured their browsers and that they are connecting to the server via Tor. It is useful to know if the user is connecting in an anonymous way to avoid network interception.

- GlobaLeaks Mobile

At the day of writing, the only project on this field is the GlobaLeaks Android application (GLDroid[8]). It is important to have the major platforms covered to enable a whistleblower to access the platform from any device in a secure way.

The Android application is in a very alpha stage; it allows connecting to a GlobaLeaks server and retrieving information, but it doesn't allow the user to send a submission yet and Tor is not fully working.

- Fax2social

This is a parallel project to GlobaLeaks to handle incoming fax, OCR them, anonymise them and submit to a third party application, like GlobaLeaks or Dropbox. The idea of using a fax phone to receive leaks is really important, especially for countries with no wide Internet availability.

[8] https://github.com/globaleaks/GLDroid

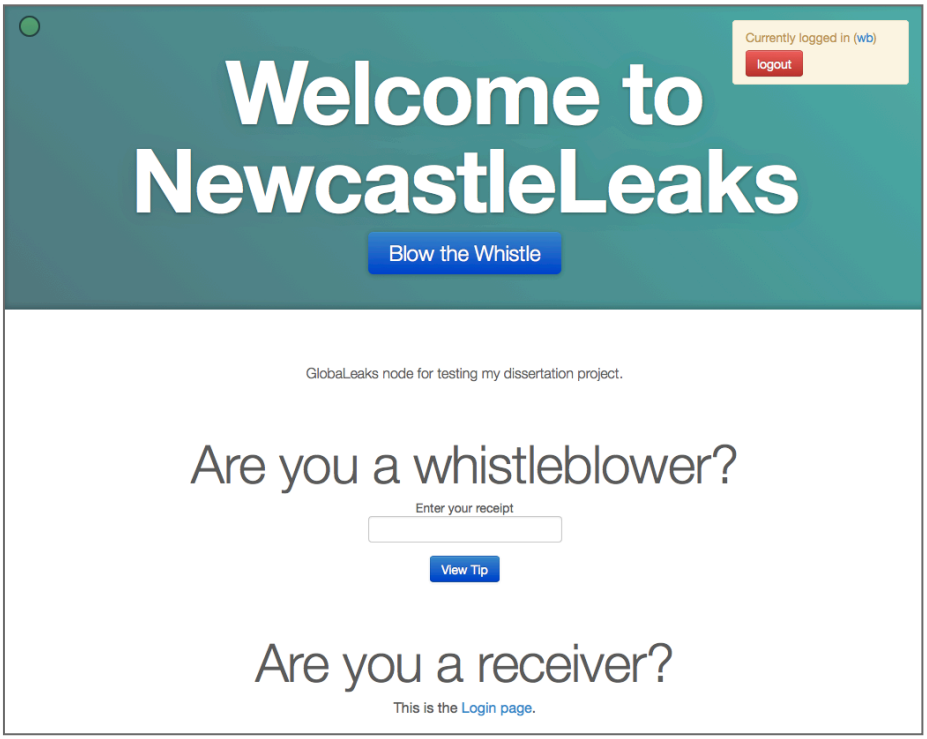

Fig. 2. An example of a GlobaLeaks' node home page

In the GlobaLeaks platform, each submission depends strictly on the deployment *context*, that is the area, category, or topic concerned by the whistleblower submission. Examples of Context are: corruption in a local municipality, environmental pollution in a specific city, human rights abuse or confidential information about something [***20***]. A context, set up by the node administrator, is defined by a short title and a description and can have one or many receivers.

A receiver is a person, who receives the submitted material and handles it, and the whistleblower when submitting it can choose which receiver should get the submission and the material.

A submission has different submission fields to fill, some of which are free text and others can be mandatory, they are set up when the context was created by the admin, the whistleblower should fill these fields with detailed answers.

A GlobaLeaks infrastructure is typically composed of three main information systems: [***20***]

- A public website to promote the initiative.
- A GlobaLeaks node to manage the whistleblowing submissions.
- A tor2web proxy to increase accessibility of the GlobaLeaks node.

2.4 Developing for iOS

As mentioned in the introduction, there is no support for iOS for GlobaLeaks at the moment, and as they aim to become the de facto standard in technologically powered whistleblowing, the support for a platform which still has more than 25% of the worldwide market share and is still the first in UK and US is really needed [***21***].

It is really important to have a whistleblowing client running on mobile phones, because it is a personal object, users tend to have fewer concerns when using it, in contrast to when using a shared computer at work or even at home.

A mobile phone follows its owner everywhere, so it is really easy and accessible in every moment For example, it would be really quick to take a picture and uploading it on a whistleblowing platform instead of going back home, transferring the picture on the computer and only then uploading it.

In chapter 2.1.3, the Honest Appalachia team mentioned the possibility of using an open wireless network or hotspot, instead of your home wireless connection. This operation will be not only quicker but also more discrete if done with a mobile phone.

When choosing to develop a mobile application there are generally two approaches: build a web application or go native [***22***]. A web application is a cross-platform solution and it works for every mobile operating system, it consists of simply restyling the original web application using a new CSS template for mobile. It is quick and supports all the mobile platforms, so why write native applications? One of the first reasons is that a native application can use the phone hardware controls, such as GPS, accelerometer, digital compass, and camera. Another reason is that a native app can work without Internet connection, which would have disabled some of its features. These two reasons played a major part in choosing which approach to follow in this project, as detailed in chapter 3 below.

When designing the different screens and tables of an app, there is a direction to follow depending on if the app is “entertainment” or “utility”. For example, compass and mail, the first one is great design, little functionalities, and it does what the user wants; the mail application, however, has a poor design by choice and any extra design elements could inhibit usability. This app, iGlobaLeaks, follows the mail app example focusing on usability and functionality and not too much on details or colour patterns.

3 Requirements Analysis

3.1 A non-technical point of view: Public Concern at Work.

Public concern at work[9] is a whistleblowing charity, which offers help to individuals and organizations providing "free confidential advices to those who witness wrongdoing or malpractice at work and are unsure whether or not raise their concern"

[9] http:// www.pcaw.org.uk

[***23***]. Their survey shows that most calls are from care or health industries (15% each) followed by education (12%) local government and financial services (9% each) [***24***].

Another interesting observation emerging from the survey is with regard to the question "To your knowledge, is there a law that protects workers who 'blow the whistle'" over 50% answered "Don't know". Similar to the Australian survey above, in the UK 87% of the people would raise a concern about possible corruption, danger or serious malpractice at work [***23***].

An interview with Public Concern at Work took place in July 2013 in their office in London to ask questions about GlobaLeaks and whistleblowing in general.

Cathy James, Chief Executive of Public Concern at Work, said that trust is an important factor in a whistleblowing action, and knowing the identity of the whistleblower increases the trust factor. Anonymity is important, but a whistleblowing platform should not be based on that, the user should have the choice on their anonymity. There is legal protection for whistleblowers that, for example, protects the identity of people who report something and then are fired from their job. In this case, they can take advantage of this law; but if the whistleblowers are anonymous, another employee can be fired instead based only on suspects and this person will not take advantage of this whistleblower protection law.

A challenge for a whistleblowing system is to have a balance between users that want to remain anonymous and people who only want to leak without anonymity to build this additional layer of trust between the user and the receiver of the submission, as some employee think it is safe and acceptable to raise the concern openly [***25***].

3.2 Technical requirements

This chapter presents the requirements that need to be implemented in the application and how they have been established.

1) *Security requirements*: The app must not save any sensitive information in order to protect the users' identity and the complaint they send. The app should also try to use a secure connection, using Tor to keep the anonymity of the whistleblower and advise the user whenever this was not possible. The network calls should be implemented in background threads, and the timeout should be not a short value, because the network requests in a Tor network could be slow.
2) *Usability requirements*: The app flow should be easy and user friendly, and it should need only a few gestures or taps to the user to do the required action. The user should not be left alone and help should be provided for every action that can be unclear. Whenever an error occurs, it should be clear and easy to repair the communicative difficulty and fix the associated problem. Furthermore, feedback should continually be provided to the users that indicate what action is being performed or what is currently happening in the application.
3) *Performance requirements*: The background network calls can be complex and require significant amounts of time, and because of the presence of Tor, the user should not notice it and the app should continue to function normally, even with limited functionality. The performance of the app with

different network speeds will be evaluated and improved during the entire development phase of this project. A cache can be used to improve the performance, but the user should have control whenever use it or not.

4 Design and Implementation

The implementation was carried out based on the analysis outlined in the previous chapter. In this chapter, all the main characteristics of the iGlobaLeaks implementation will be illustrated, starting from the design of the app and the implementation of its basic features to the documentation of the GlobaLeaks backend API, and finally to an overview of Tor applications on mobile and how to implement a Tor relay on iOS.

4.1 Design

A good practice when designing an app is to sketch out every screen with every iteration on pen and paper to have a full picture of it, maybe even creating more designs for the same screen, and after a review and a selection of the favourite screens, copy them into a mockup software to edit them in an easier way.

The first choice to make is to decide which navigation style to use. For this application a side-out panel, Facebook style, to hide the navigation buttons and have a full screen experience was selected. In this case, it is preferable not to use a Navigation controller, for example to go from the main screen to a submission, but leave all the navigation logic in the left menu, so the menu is used to decide which action to do and the right portion of the screen changes every time. Many different implementations for the slide menu have been tested, but the best graphically and the best performance is `ECSlidingViewController`[10] that provides an easy interface to implement sliding menu on the right or on the left.

[10] https://github.com/edgecase/ECSlidingViewController

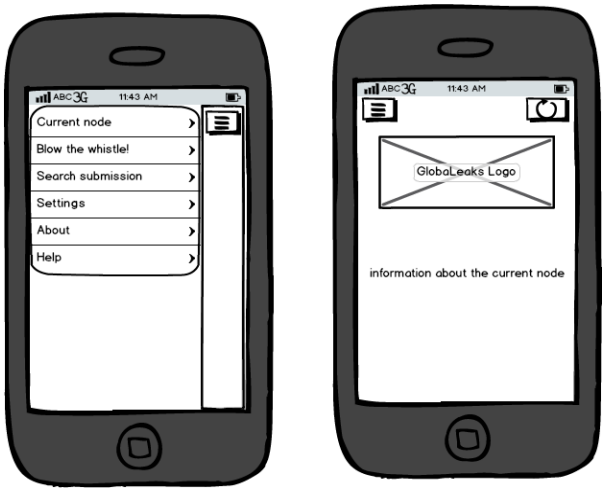

Fig. 3. First mockups designed with Balsamiq mockup. Lateral bar and main screen.

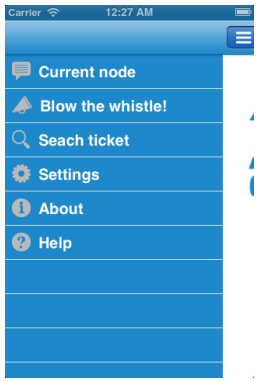

Fig. 4. Implementation of the slide-out view controller.

According to Nathan Barry [***26***], "When you first use a new app, do you know where to tap first? Questions like that determine if the app is intuitive"; therefore, the app will be improved using small icons on the left panel to pair the relative text.

4.2 Implementation of the basic features

The GlobaLeaks backend issues a receipt number for every submission, allowing the whistleblower to retrieve it and add more details or read receivers' answers; this number must be kept secret and never shared. One of the challenges of the app is how to store it without leaking it in case the phone gets in other people hands or compromised.

The first solution was to store all the receipt numbers on a Core Data database, "a powerful framework on the iOS SDK that allows programmers to store and manage

data in an object-oriented way" [*27*] in an encrypted way. It works like a sort of database where entities can be managed like objects and the framework provides methods to easy create modify and delete them.

However, this solution presents the problem that users must use a passphrase to encrypt and decrypt the database and remember it. In the event that the password and the receipt number is forgotten, users will never be able to access the submissions again, so practically, instead of remembering the receipt number users have to remember a passphrase, increasing the complexity and decreasing the quality of the user experience.

An analysis of a receipt number reveals that it is a 9-digit number that looks like a phone number, so the implemented solution aims to give the best facility to the user trading off on the security. When a submission is sent, users can choose to save the receipt number in their phonebook, where it would be difficult for an attacker to distinguish between a real contact and a receipt number stored with a legit name. Similarly, when users want to retrieve a submission, they can manually enter the receipt number or search in their phonebook. This has been implemented according to iPhone-style, so the user will be prompted to choose if creating a new contact or adding the receipt number to an existing contact according to the Apple specification example[11].

According to the analysis outlined in the previous chapter, the connections can be slow, and it can be frustrating for the user to have to wait. The application is designed to download node information as soon as the user opens it. The iOS SDK helps us with two solutions to create a non-blocking method, using an asynchronous connection or using multiple threads. Both solutions have been tested, and so far, multi-threading performed better on both 3G and Wi-Fi networks. But this is not only the factor that influenced the programing decision, as a class, GLClient, has been created to manage all the connections with the backend and returns the response, using an asynchronous connection. The client should have sent a global `NSNotification` to the app when the notifications arrives, while in the main thread there was a listener for notification, used to launch methods to updated the fields, stored the data, etc. In the actual application, using a different thread to start a synchronous connection, the response occurs as soon as it arrives while the main thread is doing other operation, mainly managing the UI.

The other implementation choice is how to implement the cache, even in this case Objective C and the iOS SDK offer various solutions, but firstly, it is important to outline the exact behaviour of the cache. We need a cache that stores little pieces of information, node or context information, stored in a JSON array; this information is not sensitive or private as GlobaLeaks nodes are freely accessible in the Internet. The cache must have a time length, and after that, it is no longer valid and must be refreshed. Firstly, consider the `NSCache` object, a collection-like container, or cache, that stores key-value pairs; this object makes a great use of the memory as it releases an object if there is low memory available, but an accurate analysis shows that this is not what we want because it is used mainly to store large quantity of data, doesn't implement a timeout and it is released when the application quits.

[11] http://developer.apple.com/library/ios/#samplecode/QuickContacts/Introduction/Intro.html

Then, the "Apple approach" was tested adding a `NSURLRequestCachePolicy` [*28*] to our connection object, the cache policy specifies if the cache should be used and how long is valid and also uses the server-side expiration HTTP header. This is a really good approach to solve the problem although the cache doesn't persist if a user closes the application, so it was important to look for the perfect solution.

Finally, the solution arose; `NSUserDefault` that stores data locally to a .plist file; this data will be deleted once the app gets deleted from the user's device [*27*]. A simple algorithm was used to store the JSON array and the relative timestamp of when it was downloaded. When a request is made, the client first checks to determine if the information timestamp and the timeout interval are greater than the actual timestamp. If it is, this starts a new connection, and if not it uses cached information as they are still fresh. This approach saves information on the device memory and is persistent between a launch of the app and another. In this case, the basic function of the app will be available (for a limited time) when there is no connection. Another workaround needed with this method is to reset the cache whenever the user changes the node URL in the settings. The refresh button on the main screen forces the refresh of the information, even if a cached version is present.

The settings panel for every iOS app was originally designed to be under the Settings app of the iPhone, but this requires closing the app, opening the Settings app, changing the relative settings and reopening the app. This flow is good for an application where changing settings is not frequently used and they do not affect several screens and some users who do not need them will never know about it.

Nathan Barry [26] points out a list of questions to ask oneself when deciding where to implement the settings of an application:

•How often will this setting be changed?

•Does it directly affect an element on a specific screen?

•What portion of your users will use this setting?

In the GlobaLeaks iPhone app, the settings will be used to change the GlobaLeaks server, determining the use of a Tor and the cache size. Those are dynamic settings that require a reload of the screen, and it would be really uncomfortable for the user to quit the app to change them; the best practice in this case is to implement the settings panel into the app and reload the screens and the connections whenever the user changes them.

4.3 Communication with the GLBackend and API

The Application Programming Interface (API) of GlobaLeaks is the component that specifies how the other software components should interact with each others, also changed much during its development phase. As this iOS application needs to communicate with GLBackend for every action, an updated version of the API documentation is needed. Unfortunately, the last updated was older than five months ago, so one of my task was to provide an updated version of this document. To get all the API calls, an entire reading and comprehension of the GLClient and GLBackend was needed, and the Android application was needed, as most of them were undocumented or changed since the 0.1 GlobaLeaks version.

First thing to do is to cut down the interactions between the backend and the client into actions, then for every action it was important to define all the communications to send and receive to achieve the desired output.

Here is a list of actions the app should perform to send submission to a GlobaLeaks node:

- Retrieving /node information
- Retrieving /contexts
- Retrieving /receivers
- Uploading a file
- Submitting submission fields
- Finalizing the submission

The first three steps are straightforward and self-explanatory. The app should send three GET request to the server to retrieve the node general information, the list of the contexts with every associated field and properties and the list of receivers for every context. An example of the request to send and the JSON array returned for every request can be found in Appendix B.

To have a responsive and user-friendly application, those calls will be made in background when the application starts, as described in the chapter above. The user will see a spinning wheel and some text on screen that informs the data is still being loaded. A background thread will take care of the download of those three basic requests, store the relative array in the cache and display on screen. Once finished, the wheel will stop spin and a reload button appears.

Once the app has all those information displayed to the user, the user can start compiling his or her submission; what is important here is the app. Before sending the submission, the programming will check that the required fields are not empty and the files were uploaded correctly, if not give an error message.

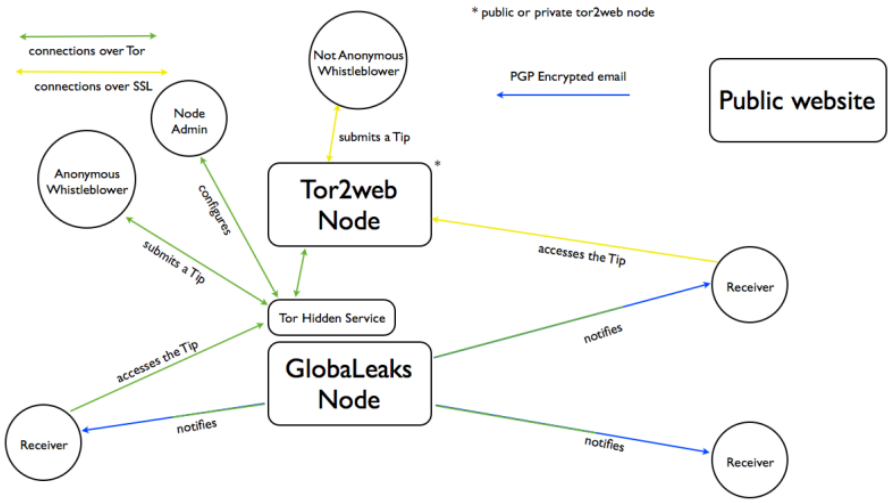

Fig. 5. All possible connections of a GlobaLeaks infrastructure

Figure 5 above represents a schema of all the GlobaLeaks' connections and links between its components; every connection can be secure (yellow link) using SSL or both secure and anonymous (green link) using Tor. The blue links show the dispatch of a PGP encrypted email from the GlobaLeaks node to the receivers, always using a Tor connection.

4.4 The submission system

The first step is to prepare a submission is to send a POST request with a submission JSON object with the context id (context_gus) and the Boolean "finalize" set to "false". The backend will answer with a JSON array with the submission fields empty and the submission id, this will be used every time we want to update the submission, or add files. Example of a submission object returned:

```
{
   "wb_fields":{},
   "pertinence":"0",
   "receivers":[],
   "expiration_date":"2013-08-11T16:02:47.223001",
   "access_limit":42,
   "receipt":"",
   "context_gus":"70b70b4a-a3dd-4e9a-b907-06a6ffbe391e",
   "creation_date":"2013-07-27T16:02:47.223102",
   "mark":"submission",
   "download_limit":42,
   "submission_gus":"955ce8ef-2416-4674-bb4a-a1b6fbc166f2",
   "escalation_threshold":"0",
   "id":"955ce8ef-2416-4674-bb4a-a1b6fbc166f2",
   "files":[]
}
```

To upload a file the application will need to send a POST request to /submission/<submission_id>/file with three defined HTTP headers: Content-Type, Content-Length: and Content-Disposition, along with the specified file attached; when the upload is complete returns a JSON array with current file information, such as file name, size and most important the file id, used to attach in our submission, e.g.:

```
[
   {
      "name":"1375208531.jpg",
      "creation_date":"2013-07-30T18:21:38.654586",
      "elapsed_time":0.7045769691467285,
      "content_type":"application/octet-stream",
      "mark":"not processed",
      "id":"00186c18-5909-4cd3-ba48-041dc16a4741",
      "size":27130
   }
]
```

According to our requirement analysis in this implementation, the app allows the user to take a new picture or select an existing one from the phone's gallery.

To update a submission with new fields values it is important to send a PUT request to /submission<submission_id> with the updated submission object, if finalize

is set to true finalize the submission and get back the submission receipt number, this is really important as a submission needs to be finalised before the user can get the receipt number.

To simplify the algorithm, the submission is not initialised (and finalised) until the user presses on the SEND button. In case they want to add any attachment, there is the need to initialise a submission because the file upload API needs a submission id, so the app will not permit to change the context anymore. The subsequent added or modified fields will be added to the Submission object and sent when the user presses on the SEND button. In this case, the request will not be a simple POST to /submission but a PUT request to /submission/<submission id>.

The API to fetch a submission, called tip, is slightly more complex: firstly the user has to login as a whistleblower sending a POST request to /authentication using the receipt number as a password,
example: `{"username":"wb","password":"5864037396","role":"wb"}`.
The response will be a JSON array with a user_id that is used as tip_id and a session id, this last one is used to authenticate the user. A header with X-Session: session_id must be added in every of the next requests.

There are 4 types of requests in the /tip API, the basic GET /tip/<tip_id> that returns the submission object with its relative fields, then /tip/<tip_id>/receivers returns all the receivers for a specified tip with their relative access counter, finally a POST request on /tip/<tip_id/file is used to add a file to the tip similarly to the submission API. Furthermore another API call can be executed on /tip to add a comment or retrieve previous comments, it is defined relatively by a POST or GET call to /tip/<tip_id>/comments, here it is an example of a comment:

```
[
    {
       "content":"commentTest",
       "source":"whistleblower",
       "comment_id":"54bba720-ba60-4d1a-848c-d8cb1946d79e",
       "author":"whistleblower",
       "creation_date":"2013-08-06T15:47:01.877532"
    }
]
```

4.5 Tor

The final step in the implementation process was to implement a Tor client into the iGlobaLeaks app to make it connect to the Tor network. This step is needed to provide a highly anonymous connection for the whistleblower and meet the security requirement listed in chapter 3. Without Tor the connection is still confidential but not anonymous. Tor is also needed because some GlobaLeaks servers allow submissions only from clients inside the Tor network.

Firstly, the variety on Tor-powered applications on mobile devices must be explained. On the Android platform, there is a Tor-based proxy called Orbot[12], it is available on the android market and offers different ways to connect to the Tor

[12] https://play.google.com/store/apps/details?id=org.torproject.android&hl=en

network; but it needs the phone to be rooted in order to work as transparent proxy and not configure any application to send its connections through the proxy.

Orweb[13] is a Tor based browser available for Android, it implements Tor and automatically connects to the Tor network on start so all the connections will be anonymous. A similar project on iOS is called OnionBrowser[14], its source code is available open source.

At the moment of writing, there are other Tor-Based applications on iOS, mainly for secure chat, but no one of them is open source and can be deeply analysed.

There are two libraries, one for Android and another for iOS platform, to implement Tor in a mobile project, called OnionKit[15], and they aim to provide all the helpers to handle Tor connection, the proxy and tool to manage the network status. The android library is fully working and updated every week, but unfortunately, the iOS library is an abandoned project, updated 6 months ago and the author says to not rely on it because of stability issues. On the 30th of July, a new framework has been published, named Tor.Framework[16], it aims to provide tor functionality to OSX and iOS application through a simple API for developers.

Unfortunately this framework is really young and doesn't have support for iOS architectures yes it only supports OSX, an updated version is coming soon and will be integrated in iGlobaLeaks. However some tests have been done under the OSX platform and it seems to be very stable, reliable and also frequently updated as its developer updates it weekly.

OnionKit could not have been implemented due to its abandoned developing state, so while waiting for an iOS compatible version of Tor.Framework, the only solution is to manually implement Tor libraries manually using OnionBrowser as an example. This approach proved viable, but it was limited in scope and the implementation classes should be updated every update of Tor libraries. This proved to be an exhausted approach because of the intense effort needed to maintain. However it is enough to accomplish the objective of having a Tor proxy inside the iGlobaLeaks application allowing it to connect to a .onion GlobaLeaks node.

5 Testing

The app was tested against requirements; 7 types of tests were defined against the requirement analysis and for each one the relative input and expected outcome is listed, then it was important to proceeded to test the app to see it responded in the expected way. The first five of the following tests have been repeated using a tor2web proxy and under a Tor connection. See Table 1 for the summary of the testing phase.

[13] https://guardianproject.info/apps/orweb/
[14] https://github.com/mtigas/iOS-OnionBrowser
[15] https://github.com/ChatSecure/OnionKit
[16] https://github.com/grabhive/Tor.framework

Table 1. List of the tests carried out with inputs, expected and actual outcome.

Test N.	Purpose of Test	Inputs	Expected Outcome	Actual Outcome
1	To see if the app caches correctly node information once opened a second time.	Open the app within a short period and without changing the node URL.	Load the node information from the cache.	The node information was displayed correctly without have to be downloaded again.
2	To see if a submission is sent correctly.	A POST request to /submission with a basic finalised submission.	The submission JSON object with the relative submission receipt number.	The expected outcome.
3	To see if multiple pictures can be attached to a submission.	Different POST requests with files attached, then a submission object with all those file ids.	The submission JSON object with the relative submission receipt number.	Both files were uploaded in the submission as we can see from this JSON array: `"files":["00186c18-5909-4cd3-ba48-041dc16a4741", "cc0e6ca9-ae69-4957-9983-9c96ca20518a"]`
4	To see if a submission is retrieved correctly using a receipt number from a phonebook.	A POST call with `{"username":"wb","password":"5864037396","role":"wb"}` A get call to /tip/<tip_id>	The tip JSON object	The expected outcome.
5	To see if node information will be updated once node URL is changed in the settings	Change the node URL in the Settings.	The app should not display old node information but load new node data.	The cache was reset and the node information were successfully loaded from the server.
6	To see if the app displays an error message trying to send a submission to a node that doesn't accept non-tor connection using a tor2web proxy.	A finalised submission.	An error message informing that the operation is not allowed.	`{"error_message": "Resource can be accessed only within Tor network", "error_code": 37}`
7	To see if the whistleblower's anonymity is protected.	The whistleblower sends a submission using Tor	The IP address of the whistleblower and the IP address connecting to the node are different.	The expected outcome: two different IP addresses.

The XCode analysis tool, Instruments, has been used to test the correctness of the memory allocation and release. In particular, Instruments measures heap memory usage by tracking allocations, measures general memory usage, checks for leaked memory and checks for over released "zombie" elements. Vandad Nahavandipoor [*27*] suggests, "Testing the same application on an iOS device proves that the memory leaks are unique to the simulator, not the device. I strongly suggest that you run, test, debug, and optimize your applications on real devices before releasing them to the App Store"

Testing results show that no major problems occurred during this phase, but this is not enough to safely assume that the application has no bugs. There are plenty of different devices and situations, but it is not time efficient to test them all, so the decision was made to integrate a crash-reporting framework. It was determined important to use Crashlytics[17], a light, anonymous and silent crash report framework that sends a crash report when the crash occurs to their server together with the device information, such as device type, free ram memory, iOS running version and many more, so the developer can spot in which function the bug exists and the exact parameters of the crash.

6 Evaluation

6.1 Security Analysis

This chapter reports a security analysis of this project and aims to evaluate it in terms of security, confidentiality, integrity and availability.

GlobaLeaks is already a secure platform; GLBackend provides different features to protect the whistleblower's identity such as stripping metadata from uploaded files and not storing any sensitive information. It is a standard practice, implemented also by the other open source projects analysed.

Whistleblowers are expected to access a GlobaLeaks server node either as a Tor Hidden Service (Anonymous) or through a Tor2Web proxy (Confidential).

iGlobaLeaks will protect its users from being tracked forcing the use of Tor, when available, providing confidentiality and anonymity as shown in Figure 6.

Figure 7 shows the flow for a Whistleblower access the server via a Tor2Web proxy. However, this scenario doesn't provide effective anonymity and introduces a trusted intermediary. The Tor2Web server is able to see all traffic unencrypted.

[17] https://www.crashlytics.com

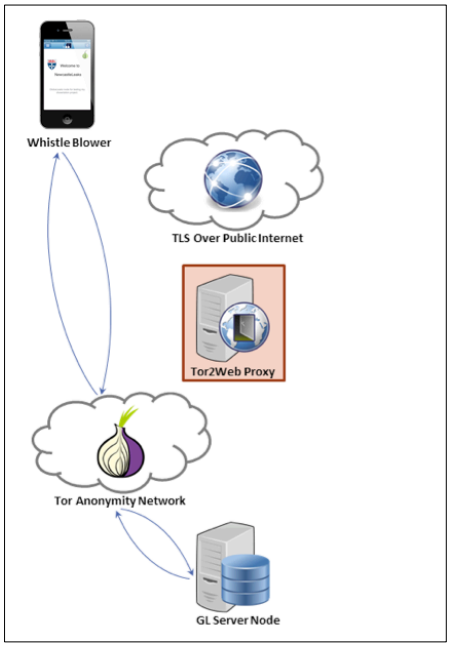

Fig. 6. Anonymous whistleblower

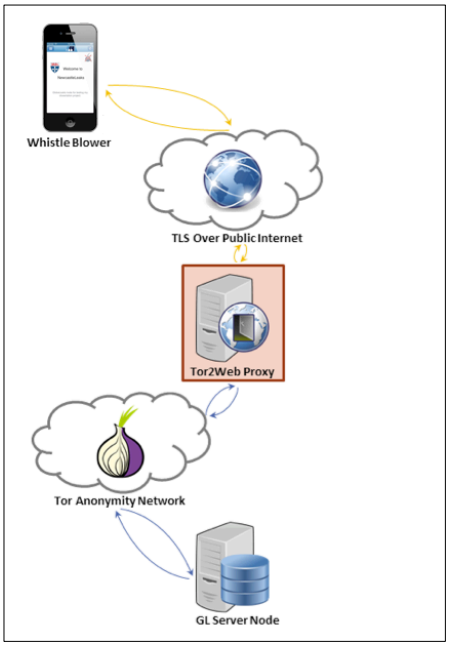

Fig. 7. Confidential whistleblower

In terms of availability the double Tor and Tor2web connection feature of iGlobaLeaks permits the whistleblower to connects to a node in two different ways, using one connection as backup if the other fails or it is unavailable.

A potential security issue was found in the function that stores receipt numbers in the phonebook. This is meant to be a user-friendly feature, however it can expose the user's privacy, in fact if an attacker manages to get the list of user's contacts, the attacker can use all the numbers present there to try to get a valid submission in the node and check what is contained in the submission. An even worse scenario can happen if the user's phonebook is linked with the user's Facebook account, in this case the attacker can get also the whistleblower's identity. A more secure method to store receipt numbers is needed in a new version of the software.

6.2 Usability Analysis

We conducted an internal questionnaire within the GlobaLeaks team about app's ease and design to get an early stage feedback. Question asked were:

- What is your role in the GlobaLeaks team?
- How do you rate the iGlobaLeaks UI design?
- Do you consider the application sufficiently user friendly?
- What and how would you improve?
- Which feature should have the priority in the further development?
- Will this application help the whistleblowing cause in your opinion?

The survey answers show how the different component of the GlobaLeaks team rated the user interface very well done and simply to use, although one of them noticed that with the vertical layout the user's attention in easily lost due to a long list of item and proposed to switch to a multi page UI. On how to improve the app they had different ideas, from an upload bar for the file transfer to add tips for the user but the next priority for most of them is to support the upload of different file types and to have an UI also for the receiver for the submission. One of them suggested to implement PGP encryption as soon as it will be implemented in GLBackend to improve the security. A unanimous answer had the last question; they all think that this application will improve making whistleblowing easier and widespread as everybody has a smartphone in their pocket.

The full answers of this survey can be found in Appendix A.

6.3 Discussion

This chapter discusses the results achieved during the five months of this project. As it has been discussed in previous chapters, the main aim of this project is to actively collaborate with the whistleblowing cause by further developing an open source software product. The iOS app of GlobaLeaks was fully developed and it is now usable, its source code is freely available on GitHub[18].

However, it is worth mentioning that not all the features of the app are fully stable, due to the alpha or beta stage of some the external frameworks that are still in development. Another issue that arose during development was that the API are undocumented at the moment, and this can be a barrier to anyone who wants to develop anything based on GlobaLeaks or simply wants to communicate with GLBackend. To solve this, a detailed API documentation has been produced and can be found in Appendix B or online[19].

Overall, not everything that was planned at the beginning was developed, for example a first idea involved the use of a plugin for the GlobaLeaks backend to evaluate the quality of the submission. The research showed how this plugin can leak some users' information and compromise their privacy and how a submission can be commented by a single receiver so the others will already be informed about the quality. Furthermore, the flexibility of GlobaLeaks contexts to create many different types of fields and check if they are compulsory or not, before the submission, helps our objective to have really detailed submissions.

However while developing the app and documenting the API some inconsistencies ad bugs were spotted, they have been reported to the relative GlobaLeaks bugs page and already fixed or waiting for a fix; furthermore some suggestions on how to solve the bugs or simplify the API requests have been provided to the main team.

[18] https://github.com/globaleaks/iGlobaLeaks and https://github.com/globaleaks/iGlobaLeaks-Tor

[19] http://docs.globaleaks.apiary.io/

7 Conclusions and Future Work

7.1 Conclusion

This project reveals positive and encouraging signals from the evolution of whistleblowing in the last ten years, not only in software approach but also in how the term "whistleblower" can become a symbol of honourable democracies [***24***].

This paper illustrates how to evaluate an open source product and make a research about similar software; then the paper analyses an open source project, GlobaLeaks, and formulates a plan to improve the frontend part by writing an iPhone application. The process of designing a mobile application is demonstrated from the early mockups to the final implementation; furthermore, the existing GlobaLeaks backend and frontend are analysed to summarise the API calls for other developers. Then, the paper illustrates how to integrate external frameworks into an existing application to improve their features, like adding the Tor support or a crash analytic service.

The testing process and the evaluation survey are also summarised, and the entire project and its future development are evaluated.

7.2 Future Work

As mentioned above, GlobaLeaks aims to become the de-facto standard in technologically powered whistleblowing, so the first priority is to have a fully stable and secure platform. Another goal is to improve the ease of the installation procedure of the whistleblowing platform for non-technical people with the goal of having a one click software installer with a bundle of GlobaLeaks, Tor and all the libraries.

For iGlobaLeaks, the priority will be to have a stable and secure Tor platform as the one implemented is pretty good at the moment; although, further research needs to be performed on the application performance under Tor network and using the Tor2web proxy. To improve the HTTP requests to the GLBackend, a framework can be implemented such as `ASIHTTPRequest`[20]; it is an easy wrapper optimised to send network request in iOS and OSX SDK with some tools to simplify the preparation of POST requests. A performance analysis of the current iGlobaLeaks software is needed to check out where and how the app can be improved.

Different kinds of files should be attached to a submission such as phone contacts, audio notes or current location; the appropriate framework will be examined and an implementation will be provided in a future release of the application.

Acknowledgments. I would like to thank my supervisor, Dr, Budi Arief for his guidance, supervision, assistance and encouragement throughout the module, which has provided much contribution to the completion of this project. Additionally, I would like to thank my parents that motivated me to study abroad and supported me through all of the years of my study. I would also like to extend my gratitude to all the GlobaLeaks team for their support, help and assistance offered. Thank you.

[20] https://github.com/pokeb/asi-http-request

Bibliography

[1] Janet P Near and Marcia P. Miceli, "Organizational Dissidence: The Case of Whistle-Blowing," *Journal of Business Ethics*, no. 4, pp. 1-16, 1985.

[2] Peter B Jubb, "Whistleblowing: A Restrictive Definition and Interpretation," *Journal of Business Ethics*, no. 21, pp. 77–94, 1999.

[3] Transparency International, "Alternative to Silence: Whistleblower Protection in 10 European Countries," Transparency International, Berlin, Prevention of and Fight Against Crime 978-3-935711-44-9, 2009.

[4] K Jean Lennane, ""Whistleblowing": a health issue," *BMJ*, no. 307, pp. 670-3, September 1993.

[5] Brian Martin, "Corruption, outrage and whistleblowing," in *Research Companion to Corruption in Organizations*, Cary L. Cooper, Ed. Cheltenham, UK: Edward Elgar Publishing, 2009, pp. 206-216.

[6] Leo Kelion. (2013, July) "*Q&A: NSA's Prism internet surveillance scheme*", BBC. [Online]. http://www.bbc.co.uk/news/technology-23051248 (Accessed 16th July 2013)

[7] Glenn Greenwald, Ewen MacAskill, and Laura Poitras. (2013, June) "*Edward Snowden: the whistleblower behind the NSA surveillance revelations*", The Guardian. [Online]. http://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance (Accessed 28th June 2013)

[8] Oleg Boldyrev. (2013, August) "*NSA spy leaks: Edward Snowden leaves Moscow airport*", BBC. [Online]. http://www.bbc.co.uk/news/world-europe-23535524 (Accessed 1st August 2013)

[9] Ewen MacAskill. (2013, June) "Edward Snowden: how the spy story of the age leaked out", The Guardian. [Online]. http://www.guardian.co.uk/world/2013/jun/11/edward-snowden-nsa-whistleblower-profile (Accessed 16th July 2013)

[10] GlobaLeaks team. (2013, January) "*Project Plan*". [Online]. https://globaleaks.org/ProjectPlan.pdf (Accessed 20th June 2013)

[11] Nadeemy Chen, "Wikileaks and its Spinoffs: new models of journalism or the new media gatekeepers?," *Journal of digital research*, pp. 157-167, 2011.

[12] Sam Gustin. (2010, December) "*Columbia University Reverses Anti-WikiLeaks Guidance*" ,Wired. [Online]. http://www.wired.com/threatlevel/2010/12/columbia-wikileaks-policy/ (Accessed 15th May 2013)

[13] GlobaLeaks team. (2013) "*Leak Directory*". [Online]. http://leakdirectory.wikispaces.com/ (Accessed 10thJune 2013)

[14] Volker Roth, Benjamin Guldenring, Eleanor Rieffel, Sven Dietrich, and Lars Ries, "A Secure Submission System for Online Whistleblowing Platforms," Cryptography and Security, Freie Universitat Berlin, FX Palo Alto Laboratory, Stevens Institute of Technology, 2013.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, vol. 13, 204.

[16] Susan Svrluga. (2012, March) "*Appalachia gets a WikiLeaks-type site*", The Washington Post. [Online]. http://articles.washingtonpost.com/2012-03-04/local/35449147_1_wikileaks-secret-diplomatic-cables-appalachia (Accessed 16th July 2013)

[17] Sue Strugis. (2012, January) "*Honest Appalachia: A Local WikiLeaks?*", Yes Magazine. [Online]. http://www.yesmagazine.org/people-power/honest-appalachia (Accessed 16th July 2013)

[18] Electronic Frontier Foundation. "*What is a Tor Relay?*", EFF. [Online]. https://www.eff.org/torchallenge/what-is-tor (Accessed 20th July 2013)

[19] Jacqui Cheng. (2008, December) "tor2web brings anonymous Tor sites to the "regular" web", arstechnica. [Online]. http://arstechnica.com/uncategorized/2008/12/tor2web-brings-anonymous-tor-sites-to-the-regular-web/ (Accessed 20th July 2013)

[20] GlobaLeaks team. (2012) "*Guidelines on how to start up whistleblowing Initiatives*" [Online]. https://docs.google.com/document/d/1Y5h_lSZq-MsefH1LkEjHN5uOfEccyyI4oJgd7tm2Iho/pub (Accessed 18th June 2013)

[21] Michael Oleaga. (2013, July) "*iOS vs. Android Market Share*", Latinos Post. [Online]. http://www.latinospost.com/articles/23701/20130716/ios-vs-android-market-share-rises-worldwide-apple-inc-s.htm (Accessed 18th July 2013)

[22] Alasdair Allan, *Learning IOS Programming: From Xcode to App Store*, 3rd ed., Rachel Roumeliotis, Ed. Sebastopol, CA: O'Reilly, 2013.

[23] Public concern at work, *Where's whistleblowing now?* London, United Kingdom: Public concern at work, 2010.

[24] Public concern at work, *Whistleblowing: beyond the law*. London, United Kingdom: Public concern at work, 2011.

[25] British Standards Institution, *Whistleblowing arrangements Code of practice*, 1st ed. London, United Kingdom: British Standards Institution, 2008.

[26] Nathan Barry, *The app design handbook*, Nathan Barry, Ed.: Nathan Barry, 2012.

[27] Vandad Nahavandipoor, *iOS6 Programming Cookbook*, 1st ed., Andy Oram and Rachel Roumeliotis, Eds. Sebastopol, CA: O'Reilly, 2012.

[28] Cox Jack, Jones Nathan, and Szumski John, *Professional iOS Network Programming : connecting the enterprise to the iPhone and iPad*, Jonathan Tang, Ed. Indianapolis, Indiana: John Wiley & Sons, 2012.

Appendix A – Survey answers:

- Question 1: What is your role in the GlobaLeaks team?
- Team member 1: I am one of the main developers of GLBackend and GLClient the two core globaleaks components.
- Team member 2: Android client developer
- Team member 3: I'm the Tor2web developer and GlobaLeaks system integrator and tester.

- Question 2: How do you rate the iGlobaLeaks UI design?
- Team member 1: Very well done.
- Team member 2: The vertical layout is not very usable and user's attention in easily lost due to a long list of item
- Team member 3: Very well done.

- Question 3: Do you consider the application sufficiently user friendly?
- Team member 1: Yes.
- Team member 2: It requires a multi page or multi tab UI to properly and easily complete a submission
- Team member 3: The interface looks simply to use.

- Question 4: What and how would you improve?
- Team member 1: I would improve how the progress of a file upload is visualised. Perhaps with a progress bar and some ETA. I would also improve the user interface for the visualisation of the leaks. I would also make the submission UI into a wizard type process (or at least allow the option of entering wizard mode and use that as a default).
- Team member 2: Submission form editor/viewer.
- Team member 3: It could be nice to introduce some tips and suggestions for the user.

- Question 5: Which feature should have the priority in the further development?
- Team member 1: Better file upload support.
- Team member 2: UI for receiver role
- Team member 3: PGP Integration (this feature is currently missing also on the official GLClient as we are currently missing a PGP support in JS. This will be really important to achieve end-to-end encryption)

- Question 6: Will this application help the whistleblowing cause in your opinion?
- Team member 1: Yes I strongly believe user friendly software (especially for mobile devices) makes whistle blowing more effective and more widespread.
- Team member 2: Definitely: everybody has a smartphone in his pocket and could blow the whistle also on the field.
- Team member 3: Yes, I think so.

Appendix B - GlobaLeaks v0.2 API

Table of Contents

---

1 Public API

API accessible to all the users

1.1 GET /node

Returns information on the GlobaLeaks node.

Request

Response

200 (OK)
Content-Type: application/json


```
{
    "description":"GlobaLeaks node for testing my dissertation project.",
    "maximum_namesize":128,
    "name":"NewcastleLeaks",
    "tor2web_admin":false,
    "tor2web_unauth":true,
    "maximum_filesize":31457280,
    "email":"lorenzo@primiterra.it",
    "languages":[
        {
            "code":"it",
            "name":"Italiano"
        },
        {
            "code":"en",
            "name":"English"
        }
    ],
```

```
    "tor2web_receiver":false,
    "tor2web_submission":false,
    "tor2web_tip":false,
    "public_site":"https://ymasaptkeux3qgbc.tor2web.blutmagie.de",
    "maximum_textsize":2048,
    "configured":true,
    "hidden_service":"http://ymasaptkeux3qgbc.onion",
    "maximum_descsize":1024
}
```


1.2 GET /contexts

Returns an array with all the contexts of the GlobaLeaks node. Every object includes submission fields and a list of receiver gus.

REQUEST

RESPONSE

```
200 (OK)
    Content-Type: application/json
```


```
[
    {
        "receipt_description":"This sequence here is your receipt keep good care of it as it will allow you to access your submission",
        "description":"?",
        "name":"Second",
        "receivers":[ "24602346-eb5e-4c49-bef3-0769790717f4", "63793685-b9a0-4ead-812d-4af62800017c"
        ],
        "fields":[
            {
                "name":"Short title",
                "presentation_order":1,
                "hint":"Describe your tip-off with a line/title", "required":true,

                "value":"",
                "type":"text"
            },
            {
                "name":"Full description",
                "presentation_order":2,
                "hint":"Describe the details of your tip-off", "required":true,

                "value":"",
                "type":"text"
            },
            {
                "name":"Files description", "presentation_order":3,
                "hint":"Describe the submitted files",
```

```
                    "required":false,
                    "value":"",
                    "type":"text"
                }
            ], "selectable_receiver":true,

            "context_gus":"d1c364a2-65ed-4794-9664-7edb9858e0e6",
            "tip_timetolive":1296000, "escalation_threshold":null,

            "submission_introduction":"Here you can submit your tip-off and the files", "file_max_download":42,

            "tip_max_access":42,
            "file_required":false,
            "submission_disclaimer":"Thank you for your contribution, your submission is complete"
        }
]
```

1.3 GET /receivers

Returns all the receivers of a GlobaLeaks node. Every receiver has personal information and a list of context he is enabled to.

REQUEST

RESPONSE

```
200 (OK)
    Content-Type: application/json
```

```
[
    {
        "update_date":"Never",
        "description":"", "tags":[

        ],
        "contexts":[ "70b70b4a-a3dd-4e9a-b907-
            06a6ffbe391e", "d1c364a2-65ed-4794-9664-
            7edb9858e0e6"
        ], "can_delete_submission":true,

        "creation_date":"2013-06-17T14:59:36.438000", "receiver_level":1,
        "receiver_gus":"63793685-b9a0-4ead-812d-4af62800017c",
        "name":"Lorenzo"

    },
    {
        "update_date":"Never",
        "description":"", "tags":[

        ],
        "contexts":[ "d1c364a2-65ed-4794-9664-
            7edb9858e0e6"
        ],
```

```
        "can_delete_submission":true,
        "creation_date":"2013-06-17T15:13:46.887787", "receiver_level":1,
        "receiver_gus":"24602346-eb5e-4c49-bef3-0769790717f4",
        "name":"Tom"

    }
]
```

2 Submission API

Used to create and update a submission

2.1 POST /submission

This creates an empty submission the specific context and returns the Id to be used when referencing it as a whistleblower. Id is a random 64bit integer

REQUEST

```
Content-Type: application/json
```

```
{"context_gus":"70b70b4a-a3dd-4e9a-b907-06a6ffbe391e","wb_fields":{},"files":
[],"finalize":false,"receivers":[]}
```

RESPONSE

```
201 (Created)
    Content-Type: application/json
```

```
{
    "wb_fields":{ },

    "pertinence":"0",
    "receivers":[
    ], "expiration_date":"2013-08-11T16:02:47.223001",
    "access_limit":42,

    "receipt":"", "context_gus":"70b70b4a-a3dd-4e9a-b907-06a6ffbe391e",
    "creation_date":"2013-07-27T16:02:47.223102", "mark":"submission",


    "download_limit":42, "submission_gus":"955ce8ef-2416-4674-bb4a-
    a1b6fbc166f2", "escalation_threshold":"0", "id":"955ce8ef-2416-4674-
    bb4a-a1b6fbc166f2",

    "files":[
    ]
}
```

2.2 GET /submission/{submission_id}

Returns the currently submitted fields and material filenames and size, this is the only interface giving back the complete submission status

Request

Response

```
200 (OK)
        Content-Type: application/json
```


```
{
    "wb_fields":{ },

    "pertinence":"0",
    "receivers":[
    ], "expiration_date":"2013-08-11T16:02:47.223001",
    "access_limit":42,

    "receipt":"", "context_gus":"70b70b4a-a3dd-4e9a-b907-06a6ffbe391e",
    "creation_date":"2013-07-27T16:02:47.223102", "mark":"submission",


    "download_limit":42, "submission_gus":"955ce8ef-2416-4674-bb4a-
    a1b6fbc166f2", "escalation_threshold":"0", "id":"955ce8ef-2416-4674-
    bb4a-a1b6fbc166f2",

    "files":[
    ]
}
```


2.3 PUT /submission/{submission_id}

Update a submission with new fields values. Returns the updated submission in a JSON array If finalize is set to true finalize the submission and get back the submission receipt number.

Request


```
{"context_gus":"d1c364a2-65ed-4794-9664-7edb9858e0e6","wb_fields":{"Full description":"22","Short
title":"11"},"finalize":true,"files":["00186c18-5909-4cd3-ba48-041dc16a4741"],"receivers":["63793685-b9a0-4ead-
812d-4af62800017c"]}
```


Response

```
200 (OK)
        Content-Type: application/json
```

```
{
    "wb_fields":{
        "Full description":"22", "Short
        title":"11"
    },
    "pertinence":"0",
    "receivers":[ "63793685-b9a0-4ead-812d-
        4af62800017c"
    ], "expiration_date":"2013-08-15T14:15:11.180227",
    "access_limit":42,

    "receipt":"9666383729", "context_gus":"d1c364a2-65ed-4794-9664-
    7edb9858e0e6", "creation_date":"2013-07-31T14:15:11.180766",
    "mark":"finalize",

    "download_limit":42, "submission_gus":"00778607-0df5-4139-b415-
    5ada02b4c98a", "escalation_threshold":"0", "id":"00778607-0df5-4139-
    b415-5ada02b4c98a",

    "files":[ "00186c18-5909-4cd3-ba48-041dc16a4741"

    ]
}
```

2.4 **POST** /submission/{submission_id}/file

Upload a file to the selected submission_id. Returns a JSON array with file information and the file id to integrate in the submission.

REQUEST

```
Content-Type: image/png
Content-Length: 27130
Content-Disposition: attachment; filename="1375208531.png"
```

RESPONSE

```
200 (OK)
        Content-Type: application/json
```

```
[
    {
        "name":"1375208531.png", "creation_date":"2013-07-
        30T18:21:38.654586",
        "elapsed_time":0.7045769691467285,
        "content_type":"image/png",
        "mark":"not processed", "id":"00186c18-5909-4cd3-ba48-
        041dc16a4741",
```

```
        "size":27130
    }
]
```

3 Tip API

Used to login and retrieve a tip

3.1 POST /authentication

Returns a session token to access the specified submission, which id was passed inside of the request password headers. And a user_id that will be used as receipt_id.

REQUEST

```
Content-Type: application/json
```

```
{"username":"wb","password":"5864037396","role":"wb"}
```

RESPONSE

```
201 (Created)
    Content-Type: application/json
```

```
{"user_id": "91a0e7c9-e704-47fb-8198-84ed707be19e", "session_id":
  "LqK8tfPKdKQF8LT8KlHBivVn9GuMTV0e66UyZmqm6y"}
```

3.2 GET /tip/{tip_id}

Returns the corrispondent submission to the tip_id sent parameter. Needs a X-Session value to be sent in the headers to authenticate the request.

REQUEST

```
X-Session: session_id
```

```

```

RESPONSE

```
201 (Created)
    Content-Type: application/json
```

```
{
    "files":[
    ],
    "pertinence":"0",
    "im_whistleblower":true,
    "context_id":"d1c364a2-65ed-4794-9664-7edb9858e0e6",
    "access_limit":42, "expiration_date":"2013-08-09T13:41:04.815883",
    "context_gus":"d1c364a2-65ed-4794-9664-7edb9858e0e6",
    "access_counter":0, "creation_date":"2013-07-25T13:41:04.815928",
    "escalation_threshold":"0", "last_activity":"2013-07-
    25T13:41:04.815928", "download_limit":42,



    "im_receiver":false,
    "fields":{
        "Full description":"this description", "Files
        description":"no files", "Short title":"This is my title"

    },
    "mark":"first", "id":"91a0e7c9-e704-47fb-8198-
    84ed707be19e"
}
```


3.3 POST /tip/{tip_id}/comments

Write a new comment for a tip. Requires the session_id and tip_id. Returns the sent comment JSON object.

REQUEST

```
X-Session: session_id
Content-Type: application/json
```


```
{"tip_id":"e78dfb77-9553-4b0d-a6f6-7b076adc62f5", "content":"commentTest"}
```


RESPONSE

```
201 (Created)
      Content-Type: application/json
```


```
[
    {
        "content":"commentTest",
        "source":"whistleblower", "comment_id":"54bba720-ba60-4d1a-848c-
        d8cb1946d79e", "author":"whistleblower", "creation_date":"2013-08-
        06T15:47:01.877532"

    }
]
```

3.4 GET /tip/{tip_id}/comments

Returns all the comments for a specified tip

Request

```
X-Session: session_id
```

Response

```
201 (Created)
```

```
[
    {
        "content":"commentTest",
        "source":"whistleblower", "comment_id":"54bba720-ba60-4d1a-848c-
        d8cb1946d79e", "author":"whistleblower", "creation_date":"2013-08-
        06T15:47:01.877532"

    },
    {
        "content":"fuffa",
        "source":"whistleblower", "comment_id":"80ed4ba4-a350-4103-b68b-
        812e0efb94e1", "author":"whistleblower", "creation_date":"2013-08-
        06T16:02:37.820577"

    }
]
```

3.5 GET /tip/{tip_id}/receivers

Returns all the receivers for a specified tip with their relative access counter.

Request

```
X-Session: session_id
```

Response

```
201 (Created)
      Content-Type: application/json
```

```
[
    {
        "name":"John",
```

```
            "tags":[
            ],
            "contexts":[ "03c28a7b-6fe4-476d-b55d-6bb3ea2fca01",
                  "6b16d055-819f-4ae8-ade8-69e204afb080"

            ], "can_delete_submission":true,
            "access_counter":0, "receiver_level":1,


            "receiver_gus":"89973c09-9dd0-408c-af66-341477d1a470",
            "description":""
      },
      {
            "name":"Lorenzo",
            "tags":[
            ],
            "contexts":[ "03c28a7b-6fe4-476d-b55d-6bb3ea2fca01",
                  "f239263d-9bdc-4996-b07c-fe0d453e5354",
                  "6b16d055-819f-4ae8-ade8-69e204afb080"

            ], "can_delete_submission":true,
            "access_counter":0, "receiver_level":1,


            "receiver_gus":"292511e0-fad5-477f-8050-05bf510a682f",
            "description":""
      }
]
```

3.6 POST /tip/{tip_id}/file

Upload a file to the selected tip. Returns a JSON array with file information.

REQUEST

```
X-Session: session_id
Content-Type: image/png
Content-Length: 43271
Content-Disposition: attachment; filename="1375888134.png"
```

RESPONSE

```
200 (OK)
      Content-Type: application/json
```

```
[
      {
            "name":"1375888134.png", "creation_date":"2013-08-
            07T15:08:41.264087",
            "elapsed_time":0.36124110221862793,
            "content_type":"image/png",
            "mark":"not processed", "id":"3ccce04e-1ea4-4254-9fac-
            344378814fb4",
```

```
            "size":43271
        }
    ]
```