Convergence: Project Specification

Tolu Alabi
Zachary Butler
Martin Dluhos

February 17, 2012

1 Introduction

For our semester long software design project, our group has decided to implement Convergence, a scheme introduced by Moxie Marlinspike to replace the Certificate Authority system widely used today to verify the authenticity of Internet websites. There are a number of deficiencies of the current Certificate Authority system, which makes veryfying the authenticity of websites less trustworthy. The first problem is that it is relatively easy to establish a Certificate Authority and in consequence many of them exist today. The system is designed so that if any one of these Certificate Authorities is compromised, it can issue a false certificate for any website on the Internet. This has, in fact, already happened multiple times, which makes the need for a replacement very pressing.

Moxie Marlinspike, a fellow at the Institute of Disruptive Studies, conceived an alternative system, which he calls Convergence. Convergence is a system where trust is not centralized in the hands of Certificate Authorities, but more distributed among so called notaries. A notary is the server-side of the system, which helps the user determine if a website she is trying to access is authentic. When a user installs Convergence add-on in her browser, the browser does not force her to use a list of pre-installed notaries, as is currently the case with Certificate Authorities. She gets to choose the notaries that she trusts herself. When she then visits a website, whose authenticity she needs to verify, she receives a self-signed certificate back. The client side of Convergence implemented in her browser will ask the notaries she trusts

if they received the same certificate. If it turns out that most of them did, then she can trust the website. Otherwise, she should be suspicious about its authenticity.

There are a number of advantages of Convergence compared to the Certificate Authority system. Firstly, Convergence gives power to the user to decide who to trust instead of requiring her to trust all the Certificate Authorities that her browser developers deem trustworthy. Furthermore, if this system becomes more widely adopted, then many notaries will be established, which will make trust more distributed than in the current model. Another advantage is that the architecture of the Internet does not have to be altered for this system to be implemented. It only requires the user to install a browser add-on and a trustworthy organization to run the notary. The add-on then disables the Certificate Authority verification and starts using Convergence instead[1].

2 Assessment of feasibility

There are two main parts of the Convergence system. One part is the client side which is implemented as a browser add-on. The other part is the server side which is the notary running on a server. Given the nature of the course, we decided to implement only the server part of the Convergence system. The main reason for this decision is that the course requires the resulting program to be written in C, which is not the programming language we would use to implement the client side in any browser. Marlinspike's Python implementation of the Convergence Project hosted on Github includes documentation, which is the main source of information for our implementatation in C. [2]

3 System Design

From the documentation available on Convergence's Github account, we learned that the domain verification process is as follows:

1. The user enters a URL in the browser.

2. The browser Convergence add-on requests a certificate from the website.

3. The website sends its certificate to the user.

4. The user sends the URL and the certificate fingerprint to the notaries.

5. The notaries request a certificate from the website the user indicated in the request.

6. The website sends its certificate to the notaries that requested it.

7. The notaries compare the fingerprint of the certificate they received from the website with the fingerprint the user provided.

8. Based on the the result of the comparison, the notary sends an appropriate response to the user.

9. User's browser tallies the responses it receives and decides if it should trust the website.

We also considered another implementation of Convergence, which we believe would be more efficient. We thought a lot of time is wasted when the user requests a certificate from the website. While the user waits for the certificate to arrive, the notaries are idle. Instead, the user could request that the notary fetches a certificate from the website while she waits for her own copy. In this model, all queried notaries return the fingerprint of the certificate they received to the user, who then compares these fingerprints with the fingerprint she received directly from the website. We decided not to implement this model since we want our notary server implementation to be compatible with the current browser implementation.

4 Software development methodology

For this project, we have elected to use the prototype and extensions model. Our bare bones notary will be listening for requests that include the URL which the user wants to confirm as well as the certificate fingerprint which the user received from the website. The prototype notary will then send a request to the website, verify that the resulting certificate matches the input fingerprint, and send the result of the comparison to the user.

While this prototype suffices as a primitive notary, there are multiple extensions we would like to add which are paramount to the notary's success.

The first extension is a cache of fingerprints for frequently and recently visited websites. Without the cache the notary would waste a lot of time looking up certificates it has already seen. In this extension, we will have to consider how long we want to keep a certificate in the cache, as well as all factors that may leave the cache invalid.

Another extension, mentioned by Marlinspike in his presentation titled "SSL and the future of authenticity", is the ability of a notary to serve as a proxy, so that requests from the user can be anonymized. If we implement this extension, the user will randomly select one of his notaries to be a proxy, and then bounce his fingerprint off of this proxy to all of his other notaries, who will then send these requests back through the proxy. This way the proxy knows who the user is, but not the website she is requesting, and the notaries know what the website is, but not who the user is. Another possible use of this extension would be to let the user use multiple proxies in a given request. This allows for some parallelization since each proxy only has to send the request to $\frac{n}{p}$ notaries (where p is the number of proxies and n is the number of notaries).

One last extension that we have thought about so far is blacklist functionality. This would allow a notary to maintain a blacklist of sites that it knows to be of ill repute such as amazon.com, where 'o' is actually the Greek omicron. If it received a request for one of these sites, it would send back a new type of response, telling the user that it doesn't trust the site. This would allow the notary to prevent phishing attacks which attempt to trick the user into going to the wrong website.

When implementing these features, we will be using the top-down methodology. That is, we will start by writing the most general functions, and progress towards the most specific ones. In this way, we will be able to detect high-level design mistakes very early. Furthermore, we are not yet familiar with the implementation-level details of this project, so beginning with high-level functions that implement the processes we already understand makes more sense.

References

[1] Moxie Marlinspike. Convergence. `http://www.convergence.io`, February 2012.

[2] Moxie Marlinspike. Convergence - github. https://github.com/moxie0/Convergence, February 2012.