Table of Contents

Conclusions

Product

Let's consider if the initial goals have been met:

1. **Provide an implementation for the Grails framework**. The plugin is published in the Grails plugin portal at http://grails.org/plugin/spring-security-rest. It's stable, mature, and many users are already using it in production applications.
2. **Make core Spring Security, which only offered form-based authentication using the HTTP Session, more REST-friendly**. Spring Security REST provides clear and easy-to-use REST endpoints for authentication, logout (when possible) and token validation. It also no longer uses the HTTP session to store the tokens, but provides a variety of options to use: from database to Memcached or Redis. Users can even provide their own implementation.
3. **Offer developers a way to implement authentication and authorization in a stateless way**. JWT support is completed, and users can use it out-of-the-box with almost no configuration. They can also easily switch between signed and encrypted JWT's.
4. **Analise, design, implement and test a robust solution, stable and highly covered by tests**. Testing has been one of the core requirements for this project. The plugin has 85 unit tests, 7 integration tests and 138 functional tests running across 3 complete testing applications. Everything is running automatically on every push at Travis CI.
5. **Contributing back to the community by creating an open-source solution**. Not only the plugin is open source (licensed under the terms of the Apache License, Version 2.0), but also a successful and popular plugin, with 99 stars on GitHub, 66 forks, 22 contributors and 44 pull requests.

Considering all the above items, we can conclude that all the goals have been met.

Process

Working alone in a large and complex project like this is not an easy task. Also we need to take into consideration the fact that the requirements are needs weren't requested by anybody in particular. This hasn't been a classical software development project where a customer has some needs, and a solution is created to solve his problems.

In my case, my research across the years, talking to and with the community and real experiences at work led to the conclusion that this plugin was needed. I drove the requirements and I implemented and tested them. And the open source community validated my approach by using it in production and expressing their gratitude publicly.

Given also that I have been working on this plugin almost entirely on my spare time, it's hard to follow any methodology, even an agile one such as Scrum. Many of the practices of software methodologies are meant for teams, not for a single person. Fortunately, I already know how they work in teams after having been working for some years.

Personal

Creating an open source solution that is also useful for others is very grateful. I have done open source development before this one for many years, and I hope I will continue doing it.

Regarding the university subjects being useful for this work, making an imagination effort I would say that the programming subjects in the first course, plus some computer engineering subjects in second course, might have been useful. But being honest I would make more important my personal effort in constantly learning new things and my career development during more than 14 years.

Future improvements

The plugin is fairly stable and mature, and from my point of view is probably feature complete. However, if I were to analyse how can it be improved, I would outline 2 main areas where it can completed.

Upgrade to Grails 3

The plugin is compatible with Grails 2 applications. Grails 2 was the latest version back in 2013. However, on 30 March 2015, Grails 3.0.0 was released.

Many people is already interested in an upgrade to Grails 3, as seen in issue 233, so therefore I will work on it in the near future.

JWT Token fingerprinting

Fingerprinting tokens is the process of storing in the token additional information coming from the original request. Examples of this additional attributes are:

- IP address.
- Session ID.
- Browser headers: Agent, Accept-Language, Accept, Accept-Encoding.
- Browser information: Screen size, OS, TimeZone.

When a token is received for validation, apart from the standard checks (validating the signature if the token is signed, decrypting if the token is encrypted, etc), the attributes will be tested against the ones originally stored in the JWT.

This is an additional layer of security that prevents tampering, man-in-the-middle attacks, etc.