MTAT.03.323 Seminar on Blockchain Technology
Implementation Project Report

Team:
Artem Fliunt
Kyrylo Voronchenko

Implementation project objective

The objective of this project was to implement an integration of KSI services with Git. The basic functionality required was to be able to sign and verify Git commits with KSI.

Motivation and usefulness

Plainly using Git to develop a crucial industry-leading software solutions is likely to have issues with securing a source code while it's important to be able to claim that it has not been altered by any means.

KSI infrastructure provides a solution by allowing to store commit information in a Blockchain which makes it possible to detect any unauthorized changes to a source code while maintaining a minimal level of complication in a development process.

With a help of KSI our **git-ksi** project allows to sign every developer commit and store it in a Blockchain so later it's easy to verify whether it's the same commit it was claimed to be.

Technology used

In order to achieve the above-mentioned functionality we developed a Python 3 script using a Guardtime Catena middleware that provides a REST JSON API over HTTP for both signing, storing and verifying signatures.

KSI kindly granted us a developer access to their services which allowed us to test our project in a 'real world' conditions. Furthermore, we received an access to a Guardtime developer guide which covers the majority of KSI functionality.

Project description

Our project in essence is a wrapper around Git which allows to pipe all the git functionality through it while automatically initiating signing/verifying process when the user commits/pulls any changes to a project under VCS.

The project is open-sourced and hosted on GitHub.

Project is shipped as a single executable python executable file (both through shell and python) which makes it easy to use by either keeping it in the same folder as files to sign or use it from your host's path.

As a middleware we use Catena DB to store signatures and communicate with KSI infrastructure.

The supported functionality is (as required) an ability to sign and verify Git commits while it's we are likely to add some more features prior the presentation day (10th of January).

As our team uses solely Linux as their OS our project supports only *Nix systems (but it’s possible to make it work on Windows though).

Goals achieved

Users of our project is able to keep development flow secure and free of integrity and authenticity issues simply by using a wrapper-like solution to Git executable.

Limitations and further efforts

The main issue with using KSI with Git is a requirement to pass signature ID of a commit to other developers for them to be able to verify a commit itself. It does not add any visible security threats but is quite in impracticable. The solution would be to store these IDs in a some kind of centralized supplementary database so they would be easily accessible to the whole development team.

The second one is the fact that developer identity can’t be directly derived from a verification procedure as it allows only to get a KSI user ID of a person who made a commit. The solution would be also to keep a supplementary DB with signature ID – user ID connection to keep track of who exactly makes a changes to a project.

Also adding full support for Windows would be a good thing to do as well as tags signing and verification.

Conclusion

We received a valuable experience of dealing with industry-ready Blockchain solution while implementing a ‘real world’ application of it. We are likely to continue working on this project to bring it to a ready-to-use state.