Certificate Transparency auditing using a smart contract

Ludvig Christensen(1003919)
Supervised by Pawel Szalachowski
12/12-2018

"Who will guard the guards themselves?" - Juvenal

Abstract

This research proposes a more transparent and publicly verifiable configuration of certificate auditors from the certificate transparency project. By creating a smart contract for the Ethereum blockchain, transparency and verifiability as well as simpler communication between auditors is possible. However the system has certain flaws which are discussed. The research suggests that the proposed alternative configuration might have desirable benefits but it's a question of tradeoffs and the proposed Proof of Concept needs more work to be used in production.

Keywords: Certificate Transparency, smart contract, Ethereum, HTTPS, blockchain, SSL

Introduction

In recent years HTTPS has become essential to internet communication with a large percentage of traffic utilizing it [1]. However there exists several problems within the underlying SSL certificate system. To battle the flaws existing within the SSL ecosystem Google started the Certificate Transparency project. Google's project combats the flaws by monitoring and auditing SSL certificates in near real time. Certificate Transparency provides the tools needed to detect SSL certificates that has been mistakenly issued or maliciously acquired as well as detect certificate authorities that has become malicious [2, 3].

Certificate Transparency has three main components. Certificate logs that keeps certificates in an append-only public log and provide cryptographic proofs of the logs properties. Certificate monitors which audit the certificate logs for potentially mistakenly issued or malicious certificates. Certificate auditors that verify the cryptographic properties of logs to make sure they are behaving according to the standard [4].

This research paper explores the benefits and pitfalls as well as implements some of the certificate auditor functionality in the form of a smart contract for the Ethereum blockchain. By putting parts of the auditor logic on the Ethereum blockchain more transparency and verifiability can be achieved since the blockchain is publicly readable and append-only. It also includes benefits in the form of eliminating parts of the need for peer-to-peer(p2p) communication between auditors and still keeping the system somewhat decentralized.

Methodology

The research has been focused of creating a Proof of Concept(PoC) with parts of the functionality of certificate auditors described in the Certificate Transparency project's website [4] and RFC 6962 [3]. The PoC consists of a smart contract written in the Solidity language, a Python script demonstrating the functionality of the smart contract and Ganache [5] for simulating the Ethereum blockchain.

Solidity [6] is an object-oriented programming language used for developing smart contracts for Ethereum and some other blockchains. It is statically typed and has features such as inheritance, libraries and user-defined structures [7]. Solidity was chosen for this research

because it is the most common language for writing smart contracts and tends to have more documentation and example code online.

A Python script is used to compile and publish the contract code on the ganache simulated blockchain as well as test calls for functions in the contract. Python was chosen for ease of use and being a language with good support for Ethereum development because of the web3.py [8].

Ganache[5] is the Ethereum test environment used in this research. It runs a local and private Ethereum blockchain for testing purposes. Ganache was chosen for ease of use.

Ethereum was chosen because it's the most popular[11] cryptocurrency with support for smart contracts. It therefore has an active development community and features being added by the day.

Results of findings

The outcome of the research is PoC in the form of code demonstrating the feasibility and properties of implementing parts of a certificate auditor in a smart contract on a blockchain. The code results are to be published on GitHub(see appendix 1).

The PoC successfully demonstrates the following certificate auditor features; verify consistency between two STHs using provided proof and verifying STH signatures. Using these two features the auditor contract is able to detect log misbehaviour if someone provides valid data to prove such an event. If a log is misbehaving it will be marked in a public log entry in the smart contract. The fundamental characteristics of the Ethereum blockchain proves to interested parties that the code was correctly executed with corresponding data and therefore makes the public log transparent and publicly verifiable.

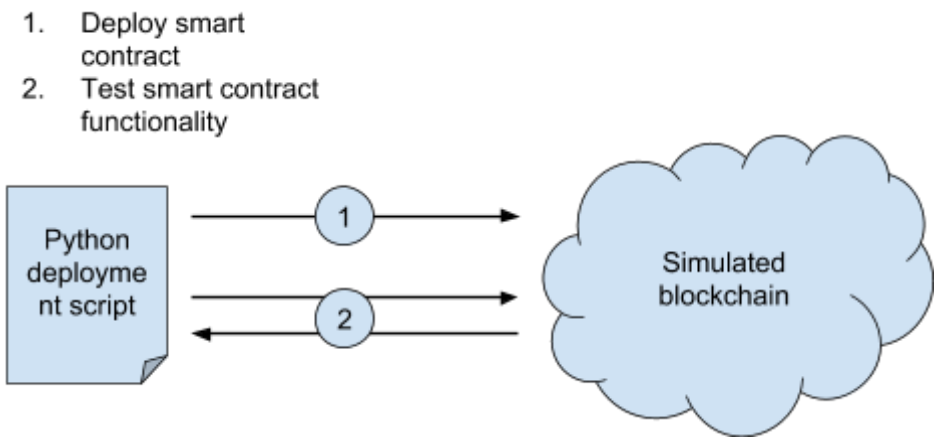

Figure 1. The PoC architecture

The overall layout of the PoC can be seen in figure 1. The Python script first compiles and deploys the smart contract onto the simulated blockchain. When the contract is deployed the Python script tests contract functionality through a series of transactions and calls which the contract responds to. The PoC is designed to run locally on a machine and not over a network.

Analysis

In this section the benefits and pitfalls of the developed PoC will be discussed as well as discussion of some of the choices made during the research and development.

The main problem this research and PoC development has targeted is the issue of communication of log misbehaviour in an efficient, transparent and secure way. How does one auditor tell interested parties, such as other auditors, monitors, certificate authorities(CAs) or website visitors, that a log is misbehaving? The Certificate Transparency RFC document proposes gossip between clients to spread the knowledge of a misbehaving log [3:Section 7.3]. This research and PoC provides an alternative to the use of a gossip protocol [9] or p2p communication. The proposed alternative creates a centralized service on top of the decentralized blockchain(see figure 2). The service both provides clients with a list of logs that has been proven to misbehave as well as accept proofs of misbehaving logs.

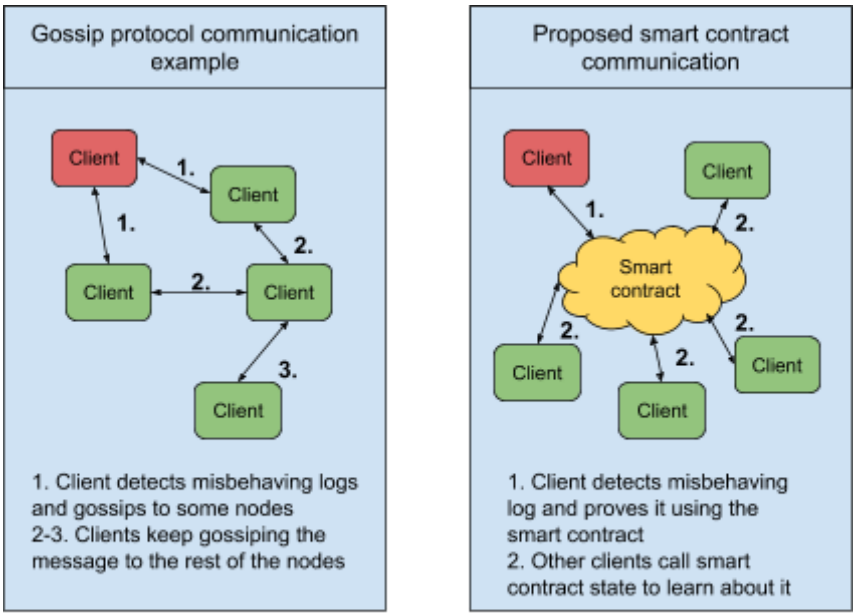

Figure 2. Gossip communication and proposed smart contract

By utilizing the transparent and publicly auditable nature of Ethereum smart contracts the proposed alternative makes sure that the proof of misbehaviour cannot be fraudulent and that everyone using the service can audit the code to prove this. The published contract code is immutable by the owner and its integrity is therefore reassured. Another benefit of the Ethereum blockchain is the redundancy gained from the decentralization. The redundancy keeps the service censorship free to a large degree, it would take a powerful adversary to take down the Ethereum network.

The alternative communication model proposed could also benefit the overall certificate transparency system by making the communication of log misbehaviour easier in regards of messaging complexity. Instead of using a gossip protocol with more complex propagation only one message is necessary to mark a log as misbehaving.

Using the alternative model has certain trade-offs. Deploying and using a smart contract on Ethereum is not free, the contract deployment transaction and subsequent proof of misbehaviour transactions must be paid for by the sender. Furthermore another trade-off is the trust that has to be put into the Ethereum blockchain. Ethereum is still a new technology that was launched in 2015, making certificate transparency an older technology. The future of the Ethereum ecosystem, and cryptocurrencies in general, is therefore somewhat unclear. If users of Ethereum, such as miners, start abandoning the blockchain the smart contract could end up in an unusable state.

Another drawback of the proposed system, somewhat connected to the issue with paying for the smart contract, is the general initiative of actually auditing the logs using the smart contract. Without proof of misbehaviour provided by some clients other clients won't learn about the misbehaviour rendering the smart contract without most of its functionality. However initiative is always needed for certificate auditors to fill their function but the barrier to entry of run an auditor might be higher using the smart contract because of the payment problem. Maybe interested parties such as CAs continuously could use the smart contract to prove logs misbehaviour.

Lastly one problem with the PoC implementation is the authentication of which log server is which. As of now logs are only identified using the index from the list of servers [10] which in a production environment is a very weak form of authentication.

Conclusion

To conclude there are several benefits found with the implementation presented as well as multiple problems. The research shows an alternative way of implementing the auditor functionality of certificate transparency and client communication which capitalize on the properties of smart contracts and blockchains. By reaping the benefits of centralization, such as ease of communication, but remaining decentralized because of the Ethereum blockchain lets the proposed system get *the best of both worlds* to some extent. However the system's value will come from the usage as it requires active auditors to submit their log views for detection of misbehaviour.

In the end it's a trade-off between the gains and losses of a system like the proposed PoC in relation to the system configurations proposed by the authors of RFC6962[3]. The research has, even though it's not the main focus, explored the practical use of blockchain technology.

Future work

Future work would logically be to work past the PoC stage of the proposed system. An experimental yet more practically functional version could be created. In such an improvement authentication of the log's identity to prevent misuse should be introduced. Such authentication could be made using the logs' public keys and validating that the proof of misbehaviour actually was signed by the log. Furthermore an optimization of the Solidity

code would help to mitigate the problems associated with costs of usage. Lastly research into putting more certificate transparency onto the Ethereum blockchain could be done.

References

[1] Statoperator, “HTTPS usage statistics on top 1M websites”, 2018. [Online] Available: https://statoperator.com/research/https-usage-statistics-on-top-websites/ [Accessed 12/12-18]

[2] “What is Certificate Transparency?”. [Online] Available: https://www.certificate-transparency.org/what-is-ct [Accessed 12/12-18]

[3] B. Laurie, A. Langley and E. Kasper. “RFC6962: Certificate Transparency”, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6962

[4] “How Certificate Transparency works”. [Online] Available: https://www.certificate-transparency.org/how-ct-works [Accessed 12/12-18]

[5] “Ganache - one click blockchain”. Available: https://trufflframework.com/ganache

[6] Ethereum, “The Solidity Contract-Oriented Programming Language”. [Online] Available: https://github.com/ethereum/solidity

[7] “Solidity language documentation”. [Online] Available: https://solidity.readthedocs.io/en/v0.5.1/ [Accessed: 12/12-18]

[8] “Web3.py”. [Online] Available: https://github.com/ethereum/web3.py [Accessed: 12/12-18]

[9] Wikipedia contributors, “Gossip-protocol”, 2018. [Online] Available: https://en.wikipedia.org/wiki/Gossip_protocol [Accessed: 12/12-18]

[10] Google. [Online] Available: https://www.gstatic.com/ct/log_list/log_list.json [Accessed: 12/12-18]

[11] CoinMarketCap, “Top 100 Cryptocurrencies by Market Capitalization”, 2018. [Online] Available: https://coinmarketcap.com/ [Accessed 12/12-18]

Appendix

Appendix 1. PoC publish link: https://github.com/ludvigch/ct-solidity