Blockchain Record-Keeping with OP_RETURN

Yassine Jebbar

May 6, 2017

Abstract

In this work, we present a detailed solution for securing Service Function Chains using Bitcoin's Blockchain. We focus on the accountability of SFC, relying on the OP_RETURN field of blockchain's transactions to transmit the different messages exchanged between the SFC components, ensuring therefeore immutable logging of the interactions that occur inside a Service Function Chain.

1 Introduction

Accountability, the third A in the famous security 3 A's (Authentication, Authorization and Accountability), is a major feature that should be taken into account in any implementation of a securtiy system. Accountablity's importance does not only lay in providing records of the different exchanges between the system's components, but it does also allow identifying points and sources of failure when the system stops working as expected or suffers a performance degradation. In the SFC context, a Service Function may shut down, a Service Function Forwarder might equally struggle with forwarding packets due to an overload. Therefore, keeping records of the interactions happening within the system enusres the system is working as expected as well as its high availability. In the next section, we give a brief reminder of the blockchain's principle, then we explain in details how Bitcoin's blockchain can be used for record-keeping.

2 Record-Keeping using Bitcoin's Blockchain

Blockchain can be generally described as a distributed ledger that serves as a record keeper for transactions occuring among the different Bitcoin's network participants. By keeping records of these transactions, the peer to peer system ensures that assets (bitcoins) are not being double-spent. The general consensus protocol adopted by the blockchain allows the majority of participants to testify whether a transactions has effectively taken place or not. Eventually, a block of multiple transactions is confirmed every 10 minutes. Once a block get "chained" within the distributed ledger, it can never be changed or altered, or at least it will require taking down the majority of the computing power of the participants to change a confirmed block of the chain (51% attack). Given this strength of the blockchain of keeping records and ensuring their athenticity, we figured out a use case where the Blockchain can be depolyed in the SFC context to keep records of the interactions occuring among the SFC components, and the meassages that are being exchanged within the Function Chain.

2.1 OP_RETURN field overview [1]

In the early days of the Bitcoin, users and participants were not concerned how the underlying blockchain technology could be used in different contexts and for various use cases, as they were embracing the breakthrough of the digtal currency emergence. As it started to get popular and gained momentum, IT developers proposed applications for the blockchain outside the finance industry: Keeping records for healthcare, Legal possession of assets, Copyrights...etc. Such applications would usually require sending/receiving different kind of messages with the transaction (confirmation, acknowledgement of receipt among others). As a consequence, developers used one way or another to get the messages sent with the transaction. Back in 2013 different players in the bitcoin ecosystem were trying to include bits of information into transactions so that they could take advantage of the irreversibility of the blockchain. Imagine, for instance, you wanted to write a contract and place it in an unchangeable location that at any future date one could go back to verify it existed. You can do this by using the blockchain. You add some bits to the transaction's scriptSig value that don't alter the end result of running that script, but allow you to store information like ”I hereby declare to give asset A to address XYZ at time UNIX_TIMESTAMP”. There were even stranger ways people would add extra bits, like including it in the BTC value of an output. Some members of the community did not like this, as they saw these extra bits as polluting the blockchain. The extra bits were a network efficiency concern because more bits meant larger block chains and more of an onus on those who are running full nodes, and it was also a community consensus concern because they thought ”we all implicitly agreed to store financial data in the blockchain, which is important to everyone, but we did not agree to store data like small text messages and invoice text”. To reach a middle ground in these opposing views, the core-developers made the opcode OP_RETURN a valid opcode to be used in a bitcoin transaction, which allows 80 arbitrary bytes to be used in an unspendable transaction. Later in February 2014, the bytes count was reduced from 80 to 40 bytes. On 16 Nov 2014 there was a pull request to have the OP_RETURN size increased “back” to 80. This was then accepted (merged) on 4 Feb 2015 [2].

2.2 Constraints

Even though using OP_RETURN for transmitting (and eventually storing) the exchanged messages inside a Service Function Chain seems like a good and equally innovative option, this solution would not be implemented without any constraints to be taken into account. The network overload could be problematic, since extra data will have to be carried within transaction packets on a regular basis. As a consequence, packets' fragmentation maybe required to ensure the overall network performance doesn't suffer a degradation. Moreover, logs' storage should also be thought about carefully, and finding the most adequate storage solution for this use case might present another problem. A solution that may be possible is finding an external storage deposit that interacts with the SFC and bitcoin network via an API. The details of the proposed solutions should depend on the implementation and the used platforms.

References

[1] https://bitcoin.stackexchange.com/questions/29554/explanation-of-what-an-op-return-transaction-looks-like

[2] http://www.talkcrypto.org/blog/2016/12/30/op_return-40-to-80-bytes/