Decentralized Firmware Updates for IoT Devices

Sotiris - Alexandros Nanopoulos

Master of Science

School of Informatics
University of Edinburgh

2018

Abstract

This doctoral thesis will present the results of my work into the reanimation of lifeless human tissues.

Acknowledgements

TBD

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Sotiris - Alexandros Nanopoulos*)

Contents

Chapter 1

Introduction

1.1 Internet of Things Era

1.2 Blockchain Technology

1.3 Dissertation Outline

Chapter 2

Security and Firmware Updates

2.1 Firmware Modification Attack

2.2 Secure Firmware Updates

Chapter 3

Decentralized Technologies

3.1 Ethereum Blockchain

3.2 Interplanetary File System

Chapter 4

Código Network Design

4.1 Threat model and Security Assumptions

A crucial part in the design a decentralized application securely, is the identification of possible adversaries, their incentives and their power. Código Network has many similarities with the OpenBazaar e-commerce platform [1]. The main difference of the two is that Código has a more narrow focus on peer-to-peer (P2P) distribution of firmware for embedded devices. The combination of the Código's narrow focus and the lack of monetary transactions makes it vulnerable to attacks. This is is due to the fact that nodes that only use the system to receive firmware updates have no immediate incentives to deviate from a proper execution of the protocol (This does not apply to OpenBazaar). Only users that develop firmware have incentives to attack it. As a result, the system should be designed to protect users from malicious developers that want to infect embedded devices with malicious software.

A malicious developer is an agent in the network that will upload a "firmware" to the network knowing that the firmware is either not working or is exploitable in some way. The benefit of polluting the network with unusable software is to grief other users of the system and cause denial of service. Polluting the network with exploitable firmware comes with many financial advantages as described in the introduction. A subtle observation is that in both cases the reward to the malicious agent is either non-monetary or even if it is monetary, then the reward comes from a source outside of the network. As a result, it is not possible

[1] OpenBazaar is decentralized marketplace similar to Ebay, that uses blockchain and cryptocurrencies to perform transactions.

to model the firmware developers as rational agents and use a game-theoretic approach to enforce them to follow the protocol. Another subtle observation, is that the adversary is in a favorable position in decentralized environments compared to centralized ones. In the decentralized setting the attacker can generate multiple identities and use them to attack the system. Essentially, in any decentralized system it is impossible for a user to determine if she is interacting with one or multiple users. Additionally, the pseudo-anonymity that comes in every decentralized application makes it harder for users to trust other agents of the network.

To protect Código Network from the aforementioned adversaries it assumed that the following hold true:

1. Cryptographic primitives such as Digital Signatures, RSA encryption, AES, SHA-256, SHA-3 are secure

2. The frameworks used (IPFS, Ethereum) are theoretically secure and implemented correctly

3. The programming languages used and the libraries used to develop the system are secure (Namely: ...)

4. Nodes downloading a firmware are able to detect malicious firmware using hardware virtualization tools [Cite Tom Spink Paper] or malware classfication software [Cite David Aspinall].

5. The embedded device used is secure.

An importance observation should be made on the 4th security assumption. This assumption is both theoretically reasonable and solves an important issue for Código network. In particular, with this assumption the system is able to address the content curation issue that arises in decentralized systems. With this assumption, we assume that nodes are able to distinguish on their own whether a firmware is malicious or honest after downloading it.

4.2 Modeling Trust in Decentralized Environments

Trust in decentralized networks is an open research area that lays outside the primary scope of this project. However, as it plays a crucial role in the system, it

would be a big omission not to address it, even partially. To do so various solutions, that are widely used in practice and have some theoretical background have been considered. Before delving into the available solutions, the trust-sensitive operations of Código Network must be identified.

A requirement specific to Código Network is that, naturally, the manufacturer of the IoT system is a trusted party. As a result, as long as the manufacturer is actively maintaining the system, there exists a firmware that is infinitely trusted by nodes. Nodes are still free to choose any firmware they want but they acknowledge the risk involved in their decision. When the manufacturer stops maintaining the firmware, the firmware owners will be incentivized to seek alternative firmware developed by the community. This is the point when nodes lose their infinitely trusted firmware and need shift their trust to an unknown developer.

A naive solution to the problem is to use a star based rating system for each developer. This solution is inadequate as a malicious developer could easily create multiple accounts (Sybil Attack) and rate himself with the highest rating thus making the rating system useless. Various heuristics, such as not counting the votes of new users, could be employed to alleviate Sybil Attack. Such heuristics are easy to deploy but hard if possible at all to optimize with respect to the user experience. For this reason, we chose to avoid such design for a more robust solution.

In general, people tend to trust less people who are not committed in their actions. This the problem a traveling salesman faces in her quest to sell her merchandise. The customers trust her less, because if she sells them a bad quality product then she has practically nothing to lose. A common workaround for this problem is providing the system with a form of commitment (Note that commitment here does not mean a cryptographic commitment). In the real world the commitment comes from opening a store. In the digital world commitments can come in numerous ways such as providing the system with valid computational PoW or by burning/donating some wealth. This kind of system, is also used in Código Network with dual benefit. Firstly, it increases the trust of user in the developers and also increases the cost of Sybil attacks. Additionally, to make the system more resistant to Sybil Attack, the cost of uploading a new firmware exponentially increases for uploading new firmware within the same day.

This paragraph may be moved to security section

From the literature the following approaches were identified for addressing

trust issues in decentralized systems.

1. Web of trust: The Web of Trust is an old idea that is currently used in the PGP project. Intuitively, its a mechanism to generate a graph with vertices's representing users and edges representing the existence of trust between to users. Then trust is quantified as the number of edges required to reach from one user to an other. With this approach trust is relative. In the Código Network setting this means that different users will trust different firmware. As a result, trust should be calculated every time a node queries the smart-contract.

2. Stake as trust: Stake as trust is novel idea that on an intuitive level shares some commonalities with the Proof of Stake concept. The idea is based on the fact that a user who allocates a lot of stake as an initial investment to upload her firmware into the network is incentivized to have developed a trustworthy firmware. An advantage of this approach is that the trust users allocate to a firmware is uniquely defined for all users.

3. Trust-is-Risk [Cite relevant paper]:

Add description

From an academic perspective both stake as trust and Trust-is-Risk have a solid mathematical background. However, neither is robustly implemented and deployed into a real world application. For this project, web of trust is chosen to model trust. The reasoning behind this choice is the successful deployment of web of trust in PGP and OpenBazzar.

4.3 Código Network Architecture

Código Network is a system of connected nodes that communicate using the Ethereum blockchain and IPFS. The communications can be made either using the Código client or a custom tool. Currently there exist two client versions, that target different type of nodes, namely the users and the developers. The client is interacting with the Ethereum blockchain. Ethereum works both as business logic computation unit and as a mean of persistence storage. Figure **??** illustrates the static architecture of the system.

The first layer of the system lays on top of Ethereum blockchain. Two smart contracts were developed on top of Ethereum. The two contracts are independently

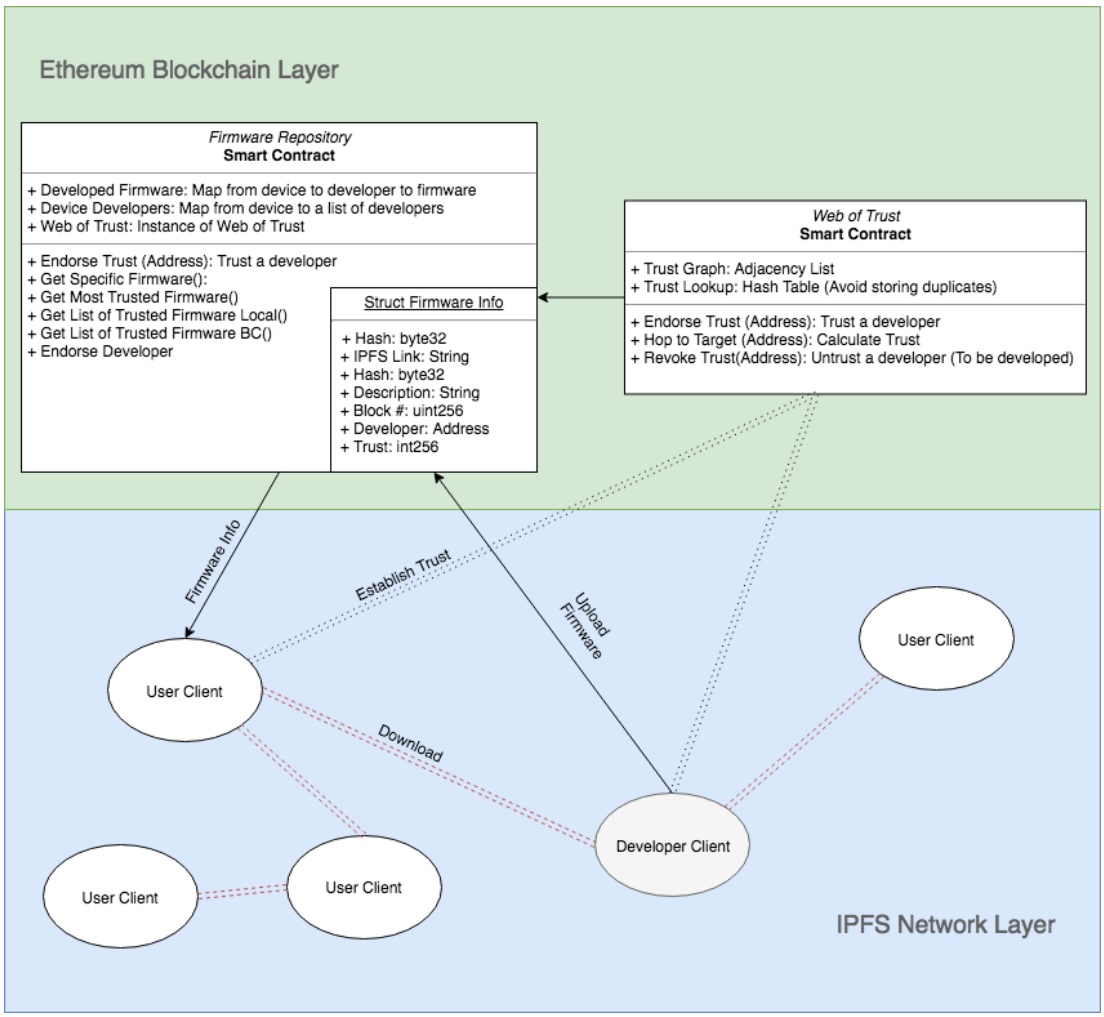

Figure 4.1: Codigo network architecture diagram

deployed and communicate with each other using Ethereum transactions. The first contract is responsible for modeling trust while the second is a firmware repository. In more depth the web of trust smart contract is an implementation of the web of trust idea in solidity. The web of trust is represented as a graph and it is implemented using an adjacency list. Nodes can use the web of trust contract independently to allocate their trust to other developers . The core function of this contract is *hop_to_target* that calculates in a recursive manner the hops required to reach from an origin vertex to a target vertex. The firmware repository can be thought of as a classic torrent website repository, that is hosted on Ethereum instead of a web-server. A firmware is represented by the following variables:

What about deal-lo-cat-ing trust?

- Firmware hash stored as 32-byte array

- IPFS link stored as a string
- Firmware description stored as a string
- Block number stored as 256 bit unsigned integer

The firmware repository interface targets both developers (want to upload firmware) and users (want to download firmware). Developers are identified by their Ethereum address. Each developer is allowed to upload two different firmware in the repository. One stable version of the firmware and one long-term-support(lts) version. Although, the number of allowed firmware per developers is two, there is no way to enforce that the stable-lts convention will be followed by the developers. Additionally, developers can use the contract interface to upload new firmware or edit the description of an existing firmware. Users can query the firmware repository to find a specific firmware, find the firmware from the most trusted developer and find a list of firmware from the most trusted developers.

To interact with the blockchain layer and download/upload a firmware from/to the IPFS network, a user and developer client were developed. It should be mentioned that the developed smart-contracts are fully autonomous and the user can interact with them with their favorite blockchain enabled tool without installing the Código client. For developers, the client automates the procedure of uploading a new firmware to both Ethereum and IPFS in a congruent way. The user client is responsible for querying the blockchain to find firmware updates and then download the latest firmware. The user-client also implements some functions that also exist in the smart-contract. This gives the flexibility to the user to choose between using Ethereum transactions to call the smart-contract, and making the calculations locally.

Chapter 5

Código Network Evaluation

Chapter 6

Conclusion and Future Work

6.0.1 Project Contribution

6.0.2 Ethical Considerations

6.0.3 Proposal for Future Work

Bibliography