University of Zurich[UZH]

INDEPENDENT STUDY – Communication Systems Group, Prof. Dr. Burkhard Stiller

Analysis of BloSS in terms of Security and Performance

Sebastian Sanchez
Zurich, Switzerland
Student ID: 17-732-710

Supervisor: Bruno Rodrigues
Date of Submission: June 20, 2019

University of Zurich
Department of Informatics (IFI)
Binzmuhlestrasse 14, CH-8050 Zurich, Switzerland

ifi

Independent Study
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuhlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

The Blockchain Signaling System (BloSS) proposes a cooperative network defense approach based on blockchain. In this regard, the goal of this Independent Study (IS) is to describe the elements that compose its architecture and perform a security analysis in the protocol that governs its operation. At a first stage, this IS decompose the components involved in its architecture, such as the Ethereum Blockchain and its smart contracts methods. Furthermore, security aspects of its off-chain storage, based on a decentralized file system (InterPlanetary File System - IPFS) are described. Lastly, a whole overview of the components present in BloSS will be detailed, from the performance results obtained to how is the reputation system implemented to work across different systems.

Contents

Chapter 1

Introduction

Nowadays, trending technologies such as Internet-of-Things (IoT) and 5G make the society prone to different types of cyber attacks. Among these attacks, Distributed Denial-of-Service (DDoS) are one of the major modern threats. A volumetric DDoS attack (*i.e.,* overloading a victim with a massive amount of network traffic) can potentially cause a disruption of the service delivery by creating an abnormal amount of traffic directed towards servers, and thus, turning these unable to distinguish malicious users and provide the requested service. In this regard, the Blockchain Signaling System (BloSS) becomes an alternative to defend in a cooperative fashion DDoS attacks [3]. The foundation of BloSS lies within the capability of a Blockchain of holding Smart-Contracts, these capability built in certain Blockchains, such as Ethereum, allows to store information using the power of the miners (in case of a Proof-of-Work implementation) of a blockchain.

Furthermore, BloSS takes advantage of IPFS which stands for InterPlanetary File System, which mimics a peer to peer system similar to a torrent client that allows sharing information across servers in a decentralized way. This preamble is the basis of how the system works externally from the created implementation. It is also worth mentioning that BloSS uses a cooperation system that will be described in detail to rank the identified IPs and distribute them across the multiple instances of BloSS.

In spite of this, is important to mention that BloSS does not focus on identifying a DDoS attack but rather to spread the information containing the IPs across servers using IPFS and the Ethereum Blockchain in the smart contract section. The underlying information about performance showed no major impact in performance with the tests presented by Rodrigues *et al.* [5].

Throughout the paper we aim to give a broad overview of how the BloSS system with a insight into topics such as IPFS and Smart Contracts. Later on, the architecture of the protocol is explained and detailed in conjunction with the reputation schema proposed by Rodrigues [4]. Following, an analysis of the protocol components will yield a security assessment in terms of stability, reliability and confidentiality taking into account performance.

This Indepedent Study (IS) is organized as follows. Chapter 2 presents related work introducing the basics on Ethereum Smart Contracts, which are the basis for the BloSS

logic in addition to an overview of the IPFS file structure, which is the off-chain storage method used in BloSS. The architecture is briefly explained in Chapter 3.

1.1 Motivation

BloSS is a innovative solution to a existing problem regarding blacklisting of IPs. It takes into account existing solutions and benefits from them providing a simple and escalable fix to identified DDoS origins. Understanding the underlying architecture of BloSS and its components which tightly relate to Reputation Schema and distribution of information in real-time can shed light upon topic such as blacklist sharing and control mechanisms in a decentralized architecture.

1.2 Description of Work

Throughout the report a full description of the components as well as in depth understanding of the underlying structure such as smart contracts in ethereum. The reputation schema proposed by BloSS and how is that peers calibrate and share across the IPFS structure.

The work will focus in achieving a full description of all the components as well as the architecture presented by BloSS.

The goal is to overview the BloSS architecture and workflow, analyzing its components and their relation, highlighting and proposing possible points of improvement/optimization performance-wise as well as possible security vulnerabilities. Thus, the following points are necessary:

1. **BloSS Overview:** involves exploring the related literature toward the comprehension of a cooperative defense and benefits of BloSS in this context. Also, such study provide the necessary basis to understand the main architectural components and their relation.
2. **Analysis of Components:** each component of the architecture must have an individual analysis as well as in the general context of the architecture in terms of performance and security, aiming the proposal of improvements in these aspects.

Chapter 2

Related Work

Among several existing works that try to reassemble BloSS we can find DOTS created by the IETF (Internet Engineering Task Force) [1] which tries to create a cooperative network that reassembles the one from BloSS but lacks the implementation in a decentralized system.

2.1 Ethereum Smart Contracts

The basis of the BloSS (Blockchain Signaling System) [6] App is Ethereum Smart Contracts, which is built to store the Hashes that will refer to IPFS (InterPlanetary File System) files containing the IP addresses of the attackers. This will be detailed in the IPFS section, whatsoever the Smart Contract lies within an important aspect of the project.

A smart contract allows to retrieve and store information regarding a transaction similar to what an object in object oriented programming works. In order for a smart contract to take place we must add gas which is a value used to insert the transaction into the Ethereum Blockchain. Once the contract is in place into the blockchain the programmed functions will allow to access the methods and return or store information, in case of BloSS initially it was used to store the IP addresses of the attackers but in a further implementation its used to store the hashes of IPFS values to retrieve the list of blacklisted IPs. In order to insert values into the smart contract we must use so called 'gas' [6] in order to be able to update the value within the contract whether it is the IP address list or the hash list. This discourages malicious users of the BloSS client to insert invalid IP lists, despite that a reputation schema was introduced which will be detailed in Chapter 3.1.

The smart contracts reassemble to normal Ethereum transactions and are bound to the same rules thus, guarantee security and immutability therefore in order to modify a smart contract they must be deleted. In order to be used by BloSS, the only information needed from each AS (Autonomous System) is the smart contract ID and enough gas to perform the insertion of data.

2.2 IPFS File Structure

The IPFS is a system that allows sharing information by using a decentralized protocol based on DHT (Distributed Hash Tables) thus, providing a list of elements to be shared across servers which only serve a reference to a block from a file. To get into context each file would get split into blocks and each of this blocks would contain a hash to serve as an ID. Each block is distributed among the peers in the network, similar to a P2P protocol in torrenting [2] and allowing to distribute block across the system to guarantee that there is always a copy in some server, thus if a single node contains the whole file then this can be used as a see to replicate over the network of participants.

In the same way, if there are duplicates within the network since files are split in equal chunks this prevents duplicating storage because the hash of the portion of code would be already within the network and unique. For this sub directory search the storing nodes use Interplanetary Name System [2] to allocate retrieve and duplicate the content in a fast manner.

Regarding BloSS, this is an ideal scenario since each node holding a AS can also hold a node of IPFS thus allowing a spread of the list across servers in case a server is down due to DDoS. During the observed results, BloSS had higher latency while using IPFS [5] but acceptable within a normal threshold. Since Bloss must add the list of the IPs to IPFS first and then generate the transaction in the Etherum Smart Contract the transaction is delayed do to IPFS and the latter is used to guarantee reliability about the published file on IPFS.

Chapter 3

Architecture of BloSS

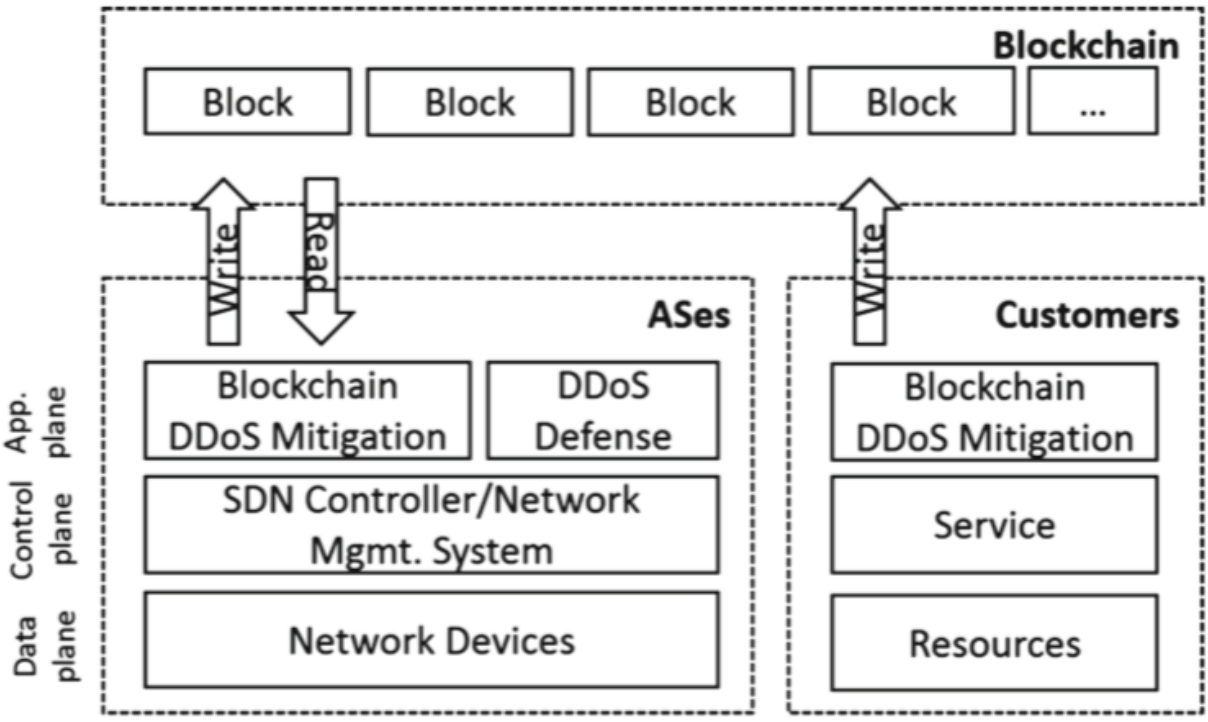

Figure 3.1: Bloss Architecture [6]

Bloss follows a workflow that is straightforward to follow. For the following, we will focus on BloSS Core even though there are other elements available such as BloSS Dashboard, Bloss Node, and Bloss Management. Here we are presented with a schema based on an Autonomous System which will manage every detected DDoS IP and add it to a list which might be added to the blockchain through a smart contractor added to the IPFS and once the Hash is calculated this will be shared in the blockchain through a smart contract. In order to keep the right balance and reliable information, a private blockchain is advised. In order to do this, a reputation schema was created that might help to mitigate this issue by providing a reputation based on the uploads and its relevance.

3.1 Reputation Schema

Among the most critical aspects of BloSS is the reputation schema which aims to take describe the protocol and a pattern that will allow a collaboration schema to set up a coordinated defense system encouraging participants through rewards or punishments and by discouraging free riders on the network. The idea behind the reputation schema is to back up the protocol, and therefore we will introduce the Target, which is the Autonomous System reporting the list of IPs that are going to be blacklisted.

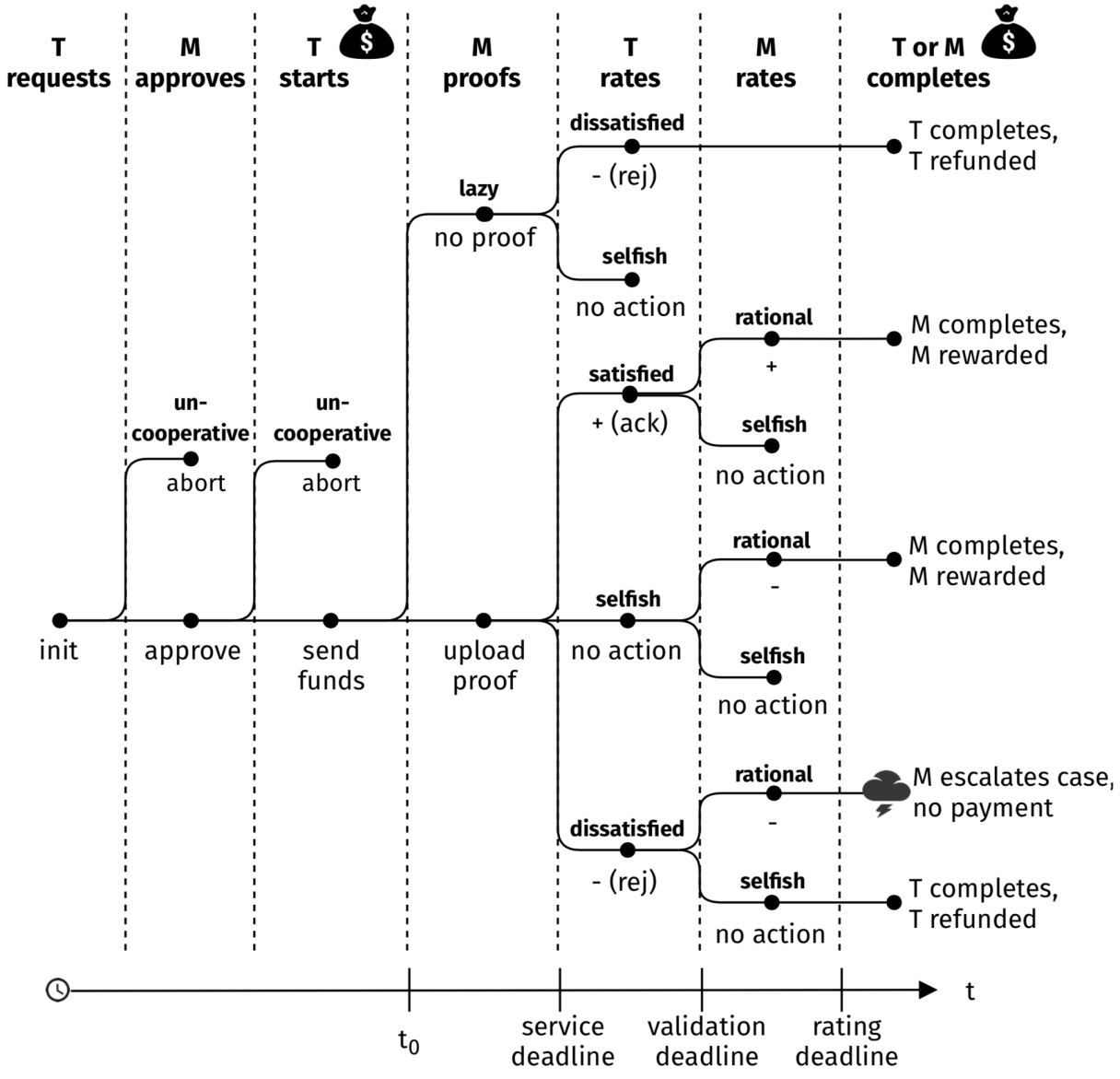

Figure 3.2: Reputation Schema of Bloss with its Protocol [4]

The process of the protocol starts by the target submitting a list of IPs, based on a reputation metric a mitigator will be selected which will implement the list and rate the blockage of the list provided. If the mitigator approves the list, then the target triggers the smart contract in order to add the list to the blockchain whether it is the IPs itself

or the hash of IPFS. Furthermore, the target will use gas to add the funds to the quick contact, or if there is no response from the target, then the process will be aborted. Once the target sends the funds through the smart contract, the mitigator must provide proof of the implementation of the target's list with its rate. In this step, if the mitigator uploads a proof, then the mitigator is capable of obtaining the reward, if no proof is provided, then the mitigator does not obtain a reward.

Assuming the mitigator did not upload a proof, this leads us to a stage where the target must rate the mitigator in which can be harmful since there was no upload from the mitigator and leads to a refund of gas or if there is no action from the target then no action is generated. This behavior is entirely unexpected since the target would only lose gas as it is shown in Figure 3.2.

On the other hand, if the mitigator does upload a proof, then is up to the target to define the further steps. As the target can be satisfied by the proof and rate the mitigator positively and therefore allowing the mitigator to rate the target in which case the mitigator is rewarded. In case of the target rates positively, but the mitigator does not rate, then anything happens because M would not get the reward, but this would not be rational behavior. In case T does not rate the mitigator's proof, then M can still rate T in which case M would get rewarded, or M might not do any action, but again this goes against any rational behavior.

Finally, if the target is not satisfied with the proof then this leads a situation in which M could escalate the case with T to understand what was the issue and until this is sorted out there is no reward for M. In the case that there is no action from M this means that T completed the cycle and M did not, therefore, T is refunded.

Is worth mentioning that each step defines a limited timespan, this starts by the moment when T uses gas to add elements to the blockchain. From there, three steps take place, which is the service deadline to check that mitigators implemented the blacklist and worked out just fine. Later on, the validation deadline aims to verify that the mitigators added the list correctly and T rates them accordingly. Finally, the rating deadline takes place after T rated the implementation from M, and now M must rate T before the end of the deadline, therefore, be eligible for the reward.

Chapter 4

Security Assessment and Potential Risks

The goal of the security Assessment will be to shed light over previously discussed matters from Rodrigues [4] and expand this into newly identified potential security issues that propose a high risk to the protocol. This will be handled from an exclusive component perspective.

Following, the protocol proposed in the previous chapter is likely to obtain good results as proven by [4]. Despite this, it is possible that there are one or more security flaws in this area related either to the protocol itself or underlying technologies within the system such as the Ethereum Blockchain or the smart contract itself.

Ballot Stuffing: There is no impediment within the reputation system or the protocol itself to avoid the two peers: mitigator and target to agree on mutual benefits over external communication channels. For example, T could agree with M on creating fake IP lists that M would verify even though the list is fake and split the reward among M and T while getting a better reputation since the reputation can be one for target and another for mitigator. In this way, both reputations go higher, and the rewards remain for the peers. [4]
Sybil Attacks: Another aspect to which the protocol is prone to error lies within the capability of the system to impersonate if the blockchain is public and therefore there is a possibility that new accounts are added and this accounts aim for their benefit rather than the networks. It is important to prevent Sybil attacks that there is an authority controlling access, this is further described in [4].
Proof Spoofing: The protocol also relies on the proof uploaded by the mitigator to the attacker. This proof might be faked or spoofed and quite hard to prove fake, in this case, there must be a way to define how proofs should be submitted and a way to test them.
Bad mouthing: In case of the Target willing to reduce the reputation from a certain mitigator, it might be producing a high amount of requests to a certain mitigator in order to reduce the chances of the mitigator of uploading a valid proof and resulting in a bad rating for the mitigator and a refund for the target.
Certificate Theft: In case a mitigator or a target is capable of obtaining a certificate from another peer in the network this would prevent knowing who is impersonating other

peer and is potentially dangerous as it could be used to elevate the rating of a particular peer without being noticed or even proved by the other peers.
Forced Timeout: A target might push hashes to the smart contract using a meager amount of gas in purpose in order to force the mitigator to miss the period given for the service deadline. This can be prevented by setting a long enough time between request and proof but relies on the capability of the blockchain miners.
51% attack on the Ethereum Blockchain: Since the Ethereum blockchain is a layer in which bloss relies upon we remain with the 51% attack issue regarding the majority of nodes. In a private blockchain, we must have only trusted nodes holding the information or else we might suffer from new nodes with altered information taking over the 'real' blockchain, thus altering the data and making BloSS unreliable. If the blockchain used is public, this makes the case rather unlikely since there is a lot of computing power and nodes in the Ethereum blockchain.

Further potential risks were found and will be addressed in Chapter 5 since they are hard to prove or are not fully proved yet. Whatsoever, those remain security issues.

Chapter 5

Discussion

Throughout the protocol, there are several aspects regarding either blockchain issues, for example, a Sybil attack towards the blockchain itself duplicating entries or modifying them according to the needs of the attacker. The protocol also lacks a way of handling escalation of cases when there is a conflict between a mitigator and a target then a third party must step in, thus making decentralization unusable.

Moreover, since there is no specific way to validate an automated proof of the uploaded IPs without trusting them; from the target side this leads to a space prone to errors or validation from the mitigator, even if the solution is implemented there's is no real guarantee besides the money discouragement that is present on the target side. Whatsoever this does not guarantee that the target will not report fake IPs within the network. Part of this project also does not take into account zombie computers that are instead part of the original clients across the network. For example, an infected attacker might also be a legitimate user of the service. If there is an interested party within the defense network that seeks economic benefits, such as blocking legitimate users from accessing services in order to gain economic advantage within the network is also a possibility and in this case the small economic impact of the target does not outage the benefit that might be obtained from blocking other legitimate clients for other peers in the network.

Additionally, since the protocol relies on an effective way to communicate the issue, this leads to impersonation problems at the communication stage. Even though is not clearly defined how is that lists are sent to the smart contract it might be through simple HTTP queries which might be prone to MITM or even SSL Stripping in which case could modify the contents of the original IP list and include different IPs in order to jeopardize the job from the original target and thus reduce its reputation.

Therefore, trust across peers in the network is essential as any AS within the network can be blocked or prevented by DDoS attacks to the communicating service to hand mitigator proofs and therefore lowering its score in an attempt to monopolize the score of a single peer. As peers within the network are aware of the details of the other AS it could follow the same direction in the fetched IP addresses that as said before cannot be distinguished from legitimate ones and therefore identify the origin of the clients in order to asses who

is the AS communicating the lists and target attacks as described above to this entities in order to gain economic advantages in the market.

Finally, the smart contract code is always prone to bugs and programming errors, and since smart contracts cannot be modified or changed, they must be destroyed to be updated. This can lead to many issues regarding security measures without a proper way to figure out who might be able to exploit the bug. Byte code is executed, but there might be attacks embedded in the byte code from peers in the network, and without constant auditing, this might become a growing issue. Versioning are control is a must in this area and logging to an external API in order to guarantee reliability.

Chapter 6

Summary and Conclusions

Finally, the study found gave an overview of the components that interact within the protocol proposed by BloSS and addressed potential security issues that might arise throughout the implementation of the protocol in a production environment. Some areas remain to be addressed in further studies but the results at the moment show flows to be fixed in an architectural level rather than the implementation itself.

Moreover, is important to take into account that the protocol is also prone to security issues found in upper layers of the protocol such as the ethereum blockchain as described in Chapter 4. To further assess the impact of this a further study would be required. At the current state of BloSS work is needed to address this issues but this can be mitigated as shown in Chapter 4. Issues identified during Chapter 5 are also to be taken into account but might require a close individual assessment to rate its importance and whether they are worth to be addressed or not.

Bibliography

[1] K. Nishizuka, L. Xia, J. Xia, D. Zhang, L. Fang, and C. Gray, "Inter-organization Cooperative DDoS Protection Mechanism," Internet-Draft,Draft, December 2016. [Online]. Available: https://goo.gl/Z2yMD8

[2] Benet, Juan. "IPFS - Content Addressed, Versioned, P2P File System." CoRR abs/1407.3561 (2014): n. pag.

[3] Rodrigues, B., Bocek, T., & Stiller, B. (2017). Enabling a cooperative, multi-domain DDoS defense by a blockchain signaling system (BloSS). In Proceedings of the 42nd IEEE Conference on Local Computer Networks.

[4] Gruhler, A., Rodrigues, B., & Stiller, B. (2019, April). A Reputation Scheme for a Blockchain-based Network Cooperative Defense. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 71-79). IEEE.

[5] Rodrigues, B., Eisenring, L., Scheid, E., Bocek, T., & Stiller, B. (2019, April). Evaluating a Blockchain-based Cooperative Defense. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 533-538). IEEE.

[6] Rodrigues, B., Bocek, T., Lareida, A., Hausheer, D., Rafati, S., & Stiller, B. (2017, July). A blockchain-based architecture for collaborative DDoS mitigation with smart contracts. In IFIP International Conference on Autonomous Infrastructure, Management and Security (pp. 16-29). Springer, Cham.