Increasing Security of Nodes of a Blockchain by Simple Web Application Firewalls

Marc Jansen
Computer Science Institute
University of Applied Sciences Ruhr West
marc.jansen@hs-ruhrwest.de

***Abstract*— In recent times, a lot of attacks against central server infrastructures have been recognized. Those infrastructures have seen attacks ranging from attacks against Internt of Things (IoT) infrastructures, via attacks against public infrastructure to attacks against cryptocurrency exchanges and blockchain based infrastructures themselves, e.g., the already almost legendary Decentralized Autonomous Organization (DAO) hack. Measured by press coverage, attacks against cryptocurrency exchanges and infrastructures seem to be among the most prominently reported attacks, probably due to the large amount of money that is stolen during those attacks and the great (but obviously still quite risky) potential (and financial involvement) of the blockchain technology. Naturally, attacks like the ones we have seen recently increase the notion of uncertainty of blockchain technologies among the people, reflected in lower values of cryptocurrencies in general. Obviously, this demands for an overall increase of security of cryptocurrency based technologies. Therefore, this paper provides an architectural approach, based on a proxy, to increase security of publicly available nodes of a blockchain based technology. Furthermore, it provides a first evaluation of the approach based on the results of an extensive community test of a new cryptocurrency.**

Keywords—blockchain; security; Web application firewall; proxy

I. INTRODUCTION

When Satoshi Nakamoto published his famous paper about a potential peer-to-peer payment system [1], the overall success of the proposed system could hardly be estimated. In 2016, bitcoin itself is dominating the cryptocurrency world by a market cap of about 12 billion $, while the overall market cap of cryptocurrencies is about 14 billion. Already those figures demand for a high security of blockchain based installations and scenarios. Blockchains are a general approach that allows to store transactional data in an audit proved way. Furthermore, there are a number of approaches that utilize blockchain technology beyond the usage of cryptocurrencies, among others, e.g., for securing intellectual property rights [2].

Nevertheless, cryptocurrencies and blockchain based technologies in general have not been widely accepted by users. Even those users already working and investing in blockchains seem to be quite suspicious, e.g., the Ethererum blockchain lost about 25% of its market cap right after the hack of the DAO platform[1]. This also supports the demand for an increase in the security of blockchain based solutions, while at the same time, blockchain based solutions provide quite easy entry points via publicly available API's in form ReST-ful Web Services [4].

[1] http://www.coindesk.com/understanding-dao-hack-journalists/

In order to provide a reasonable description of the presented approach, two terms from the domain of IT-Service Management need to be introduced here, in order to properly understand what the approach allows to do and what its (natural) barriers are. Traditionally, security flaws in distributed systems could be referred to as issues. In IT-Service Management issues are further differentiated into incidents and problems. In the IT Infrastructure Library (ITIL) framework, an incident is defined as "An unplanned interruption to an IT service or reduction in the quality of an IT service." [5], while furthermore, a problem is described as "The unknown root cause of one or more existing or potential incidents." [5]. The approach described in this paper will, according to those definitions, mostly tackle incidents and it does not try to solve the underlying problems.

The remainder of this paper is organized as follows: First, an overview of the current state of the art in the domain of securing Web applications is provided. Afterwards, the architecture developed in order to control and increase the security of blockchain based applications is described, followed by a description of the implementation done for an example blockchain, the Waves Platform. Waves provides a relatively new blockchain based technology, that allows to easily create one's own tokens. It is based on the Scorex framework, developed especially for research purposes in the blockchain domain. Additionally, an evaluation of the developed approach is presented. Finally, this paper concludes with a section that provides an outlook on future work.

II. STATE OF THE ART

A number of projects currently concentrate on the security of Web based applications. First of all, the Open Web Application Security Project (OWASP) [6] needs to be mentioned here. This project continously scans the Web for traditional and new attack vectors in order to provide a list of prominent attack vectors, even ranked by their appearance. Furthermore, the project also provides schemes and patterns for the recognized attack vectors that allow to identify malicious request and to filter those malicious requests out before they actually hit the target. Prominent examples of such attack vectors are SQL (Structured Querying Language) injections, directory traversal attacks, XSS (Cross-Site-Scripting) and/or CSRF (Cross-Site-Request-Forgery)´ attacks.

In order to provide security against identified attack vectors, different architectural approaches like WAFs (Web Application Firewalls) or proxy technologies are usually deployed. Here, e.g., NGINX as a lightweight Web Service instance has made its name in the community for being an easy to configure proxy that allows some basic filtering

 ISBN: 978-1-61208-563-0

functionality for fiddling with attack vectors as mentioned above.

A proper proxy configuration, as necessary, e.g., for NGINX, is not easy to achieve and does not provide enough flexibility to actually filter with more specialized configurations, e.g., depending on the source of the request in connection with the target endpoint. Furthermore, it does not provide rich functionalities that allow to handle possibly recognized attacks, e.g., by blocking the attacking host for a given period of time.

Additionally, using a simple WAF does also not allow filtering the access to certain Web Service endpoints in relation to the source address of a given request.

Therefore, in order to overcome the shortcomings of the mentioned approaches, this paper provides an approach that allows both, filtering access to certain endpoints, e.g., by source address, and (at the same time) allowing to also filter for known and well documented attack vectors.

III. ARCHITECTURE

In order to understand the architectural improvements implemented by the presented approach, we first have to have a look at the standard architecture of modern blockchain based implementations. Figure 1 presents an overview of an usual blockchain based architecture, consisting of an usual Peer-to-Peer (P2P) based blockchain architecture on the right-hand side in which the different nodes that participate in the network are connected with each other. On the left hand-side, the connection of a number of different clients to the nodes of the blockchain are visualized. In the upper part of the figure, a zoom to one node is presented, showing that each node actually provides two different ports to the network, one for the connection to the other nodes in the P2P network and one mainly for the connection of clients. While the port for the connection to the P2P network usually communicates over internal protocols, mostly proprietary to the blockchain, the port for the connection to the clients usually provides a ReSTful interface for the communication with the clients.

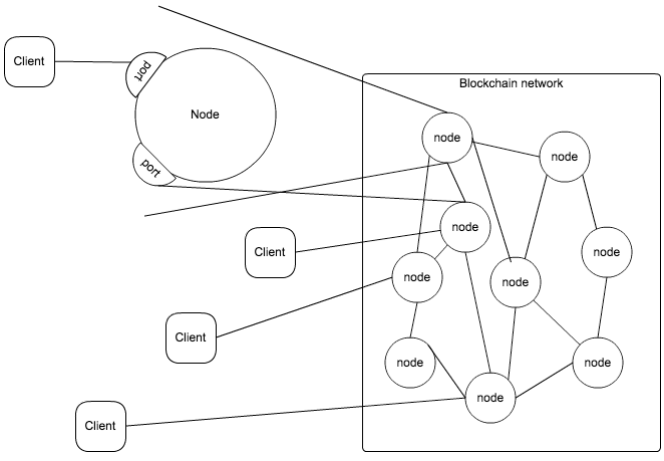

Figure 1: Basic blockchain architecture

It is important to understand that Waves uses a PoS (Proof-of-Stake) based approach, providing significant advantages above PoW (Proof-of-Work) based approaches like Bitcoin [7], especially with respect to power consumption and network fairness.

The presented approach now tackles the problem that a malicious client could try to make use of the Web Service endpoints of the node in order to enter malicious code or try to exploit issues in the node. Therefore, a proxy (according to the proxy or façade design pattern [8]) instance could be installed in front of the port for the communication with the clients in order for being able to filter against certain patterns of malicious code or restrict the access to a limited number of Web Service endpoints, e.g., necessary for the administration of the node. Figure 2 provides an overview of a node that is secured by a proxy instance.

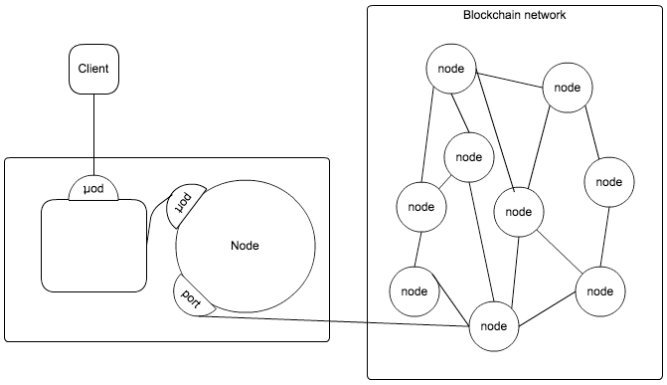

Figure 2: Architecture after the inclusion of the proxy

Internally, the proxy basically performs two different tasks. On one hand, it filters the access for different endpoints and decides if the request to a certain endpoint should be allowed or forbidden, potentially also by taking the source address of the request into account. As an additional task, the proxy could also filter for malicious code inside the request. Together, the internal architecture of the proxy could be visualized as in figure 3.

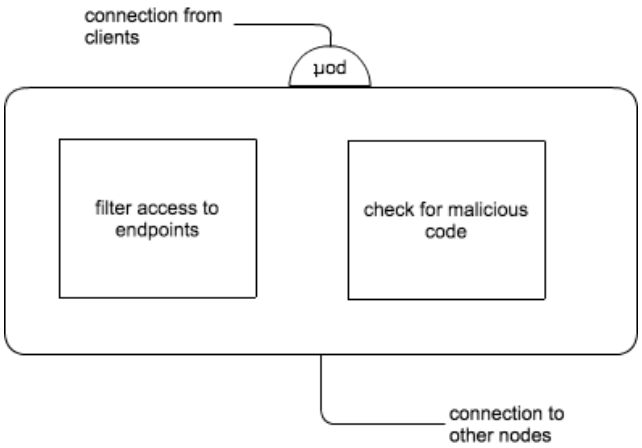

Figure 3: Internal architecture of the proxy

IV. IMPLEMENTATION

The description of the implementation first provides changes necessary to the usual configuration of a node of the

blockchain, followed by a description of the implementation of the implementation of the proxy.

A. Configuration of the blockchain nodes

In order to implement the above described architecture efficiently, some configurations on the side of the node are necessary. First of all, it needs to be ensured that the node just listens for local connections. This could usually be achieved by different means, depending on the possibilities provided by the node. With some technologies, the nodes may just be configured (via a configuration file) to listen just to the loopback interface, other node technologies might provide something like a whitelist for addresses that are allowed to connect to the node. Also, a combination of both approaches might be possible. Furthermore, it is often helpful to change the default port of the node to a different port in order to allow the proxy to bind on the default port, and, by this, to allow the node to be reachable for the clients via the standard nodes port.

B. Implementation of the proxy

The first example implementation was based on NodeJS as a server side JavaScript framework. Although any other server side programming language and environment would also be suitable, NodeJS seemed to be a quite natural choice due powerful APIs (e.g., ExpressJS [9]) for Web based solutions. Furthermore, a couple of Web Application Firewall frameworks already exist that implement the latest findings from OWASP. Therefore, the decision was made to rely on those frameworks in order to capture standard attacks like CSRF (Cross Site Request Forgery) [10], XSS (Cross Site Scripting) [11] and alike. Also, other attacks that are not necessarily possible against a blockchain node, e.g., SQL injection or directory traversal is reasonable to check in order to take countermeasures against attackers that just randomly scan networks and try to apply random attacks. Also, Web Applications Firewall usually provide certain countermeasures against attack, e.g., by putting the address of a malicious attacker on a blacklist so that further attempts of attacks are no longer possible from listed addresses. Prominent examples for these kinds of Web Application Firewall APIs that are available for the ExpressJS Web framework are ExpressWAF [12] and/or lusca [13].

In addition to the filtering for standard attacks, also filtering for specific endpoints should be possible. For this, an easy to describe JSON based configuration file allows to configure the endpoints that are allowed to access, the source address that is allowed to access the endpoint (if defined, otherwise the endpoint is openly available) and the type of HTTP request allowed to those endpoints. The following JSON file shows an example for the configuration file.

```
[
    {
        "method": "GET",
        "source": "134.91.",
        "path": "/blocks/height"
    },
    {
        "method": "GET",
        "path": "/node/version"
    }
]
```

Here, the first entry describes that the */blocks/height* endpoint is accessible from IPs in the *134.91.0.0/16* network via HTTP GET requests, while the second entry permits HTTP GET requests to the */node/version* endpoint globally.

This configuration file is parsed in at startup of the Web Application Firewall, configures itself properly and instantiates a filter method as follows:

```
var filter = function(req, res) {
    var path = req.url;
    var source = req.connection.remoteAddress;
    var method = req.method;

    filterConfig.forEach(function(filter) {
        if (source.startsWith(filter.source) &&
            path.startsWith(filter.path) &&
            filter.method === method) {
                return true;
        }
    });

    return false;
};
```

This method was integrated in the express-http-proxy module, and by this enables the proper filtering according to the rules defined in the JSON configuration file.

V. EVALUATION

In order to evaluate if the presented approach provides an added value in the sense of increased security, the approach was implemented for securing Nodes of the Waves Platform network.

A. Scenario description

The Waves network is a relatively new blockchain that was launched in the first quarter of 2016, having an easy to use token creation process in mind. The Initial Coin Offering (ICO) started in March 2016, ending by collecting about 30.000 Bitcoin, making it the sixth ever most successful crowdfunding campaign. After this successful ICO, the team provided the code for running nodes of the network, so that investors and other interested parties could participate in stabilizing the network. At the same time, especially investors have been very concerned by the question of the security of the nodes and because of that, the Waves Platform team announced a hackathon for finding bugs in the system. This led to a tremendous effort of the community for finding bugs and reporting those via GitHub [14] to the development team.

 ISBN: 978-1-61208-563-0

Therefore, this github repository provides a rich resource for evaluations.

B. Analysis

The major idea for the evaluation presented here is to evaluate the presented approach with respect to the reported issues.

In total, 54 issues have been identified and been documented in the issues section of the Waves platform github account. These are divided into 39 issues directly in the Waves Platform software and 15 issues in the underlying Scorex framework, used by the Waves Platform developers. From these 54 issues, 19 (13 in the Waves code, 6 in Scorex code) have been security related. From those 19 reported security related issues, 11 could have been solved by using the presented approach. It is important to stress here, that in IT-Service Management terms, not the underlying problem would have been resolved, but the incident of potential execution of malicious code would have disabled. Overall, this results in 57.89% of potential attacks that would have been prevented.

The low amount of reported security related issues mainly comes from the short testing period, which seemed to be appropriate due to the fact that the underlying framework was already well tested. Later evaluations can rely on larger samples.

Having a closer look, a major difference between issues found in the underlying Scorex framework and issues in the Waves Platform code become obvious. From the 6 identified issues in the underlying Scores framework, only two would have been prevented by the usage of the presented approach, resulting in 33% of potential attacks that would have been prevented. Having a look at the Waves Platform code on top of the Scorex Framework, from the 13 identified issues, 9 would have been resolved by using the presented approach, calculating to a prevention of 69.23% of possible attacks. This clearly significant difference leads to the hypothesis that issues in more basic functionalities are harder to prevent by the presented approach than issues providing more abstract functionalities.

VI. Conclusion & Outlook

The paper presented an architectural approach for increasing the security of peer-to-peer nodes of a blockchain technology where the functionality of the nodes is at least partially made available via Web Service endpoints. Those functionalities could be made available in a more secure way by providing a simple Web Application Firewall, allowing for filtering the access to certain endpoints by different aspects. As a result, it was shown that a large number of problems at higher level of abstraction could be eliminated by the approach, whereas the security of lower level functionalites could not be improved that dramatically.

Therefore, in future work, a possible aspect could be to also improve security of lower level functionalities by implementing a similar architectural approach directly on the protocol level of the peer-to-peer protocol in addition to the security increase on the higher-level Web Service endpoints. By this, similar results should also be achievable as for the Web Service endpoints. This, of course, needs to be evaluated in more detail. Another goal would be to integrate the described approach directly in the nodes in order to ensure that the security measures are taken by all nodes of the network and to decrease architectural complexity.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf”, last visited: 28.11.2016

[2] B. Gipp, N. Meuschke, and A. Gernandt, “Decentralized Trusted Timestamping using the Crypto Currency Bitcoin”, in Proceedings of the iConference 2015, Newport Beach, California, 2015.

[3] http://www.coindesk.com/understanding-dao-hack-journalists/, last visited 15th of June, 2017

[4] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures”. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, last visited: 01.12.2016

[5] ITIL Service Strategy, “Office of Government Commerce”, TSO, London, 2007.

[6] https://www.owasp.org/index.php/Main_Page, last visited 15th of June, 2017

[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “Design Pattern – Elements of Reusable Object-Oriented Software”, pp. 185-195, Addison-Wesley, 1999.

[8] https://bitcoin.org/bitcoin.pdf, last visited 15th of June, 2017

[9] https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF), last visited 15th of June, 2017

[10] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS), last visited 15th of June, 2017

[11] http://expressjs.com, last visited 15th of June, 2017

[12] https://github.com/ToMMApps/express-waf, last visited 15th of June, 2017

[13] https://github.com/krakenjs/lusca, last visited 15th of June, 2017

[14] http://www.github.com, last visited 15th of June, 2017

 ISBN: 978-1-61208-563-0