HACKEN

NODE SECURITY REVIEW REPORT

Customer:	Ambrosus
Date:	February 4, 2019
Platform:	Ethereum
Language:	Javascript

This document may contain confidential information about IT systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.
The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities fixed - upon decision of customer.

Document

Name	Node Security Review for Ambrosus
Platform	Ethereum / Javascript
Date	04.02.2019
Node Link	https://github.com/ambrosus/ambrosus-node
Node Commit	9c36109464a481e8e42de673a75590b059595553
Node Branch	master
NOP Link	https://github.com/ambrosus/ambrosus-nop
NOP Commit	79e20f5d3948f7b397e81f148a1aae04c472a25b
NOP Branch	master

Team Composition

Blockchain Security Lead	Pavlo Radchuk - PR
Offensive Services lead	Eduard Babych - EB
Application Security Engineer	Vadym Shovkun - VS
Application Security Engineer	Danil Matveev - DM
Blockchain Security Engineer	Serhii Okhrimenko - SO
Blockchain Security Engineer	Evgenii Marchenko - EM

Table of contents

Document 2
Team Composition 2
Table of contents 3
Introduction 4
Scope 4
Executive Summary 5
The distribution of Findings 7
Severity Definitions 7
Ambrosus Node and NOP Security Review 8
Security Review Findings 27
Conclusion 38
Disclaimers 39

Introduction

Hacken OÜ (Consultant) was contracted by Ambrosus (Customer) to conduct a Node Security Review. This report presents the findings of the security review of Customer`s codebase conducted between January 25th, 2019 - February 4th, 2019.

Scope

The scope of the project is node codebase and NOP scripts of Ambrosus project.

The scope of tasks performed during the project is listed below:

1. Finalize Scope
2. Setup nodes
3. Review of cryptoeconomics specification against potential threats
4. Understanding the system by using its functionality
5. Permission checks against matrix
6. Calculate collision probability - analyze to write a report
7. Auto scanning of the codebase
8. Analyze node upgradeability mechanism
9. Review NOP script and analyze potential threats

10. Dump and analyze traffic between nodes
11. Privilege escalation
12. Docker escape testing
13. Fuzzing of APIs (all parameters in GET, POST, PUT requests)
14. NoSQL injection testing
15. Testing and code review of token generation
16. Manual review of timeout mechanism
17. Manual review of auto scanner findings
18. Analysis of KYC process
19. Manual code review for immutability of data (Merkle proofs etc.)
20. Analyze potential deserialization vulnerabilities
21. Web pentest for Hermes client side
22. Network discovery + scanning of the nodes
23. DDoS simulation
24. Analyze private key storage and usage
25. Analysis of cryptography implementation
26. Report development

Executive Summary

Hacken team performed security review for the Customer system. The project focus was on 2 factors - web/network penetration testing for the deployed nodes and blockchain security assessment for node codebase and NOP script.

The scope of the work was agreed with Customer at the start of the project and the review was conducted covering the scope. The scope includes attacks on all endpoints that are simulated for 4 main classes of potential attackers:

- external attacker
- external attacker with access to API
- attacker that hosts Hermes node
- attacker that hosts Atlas/Apollo node

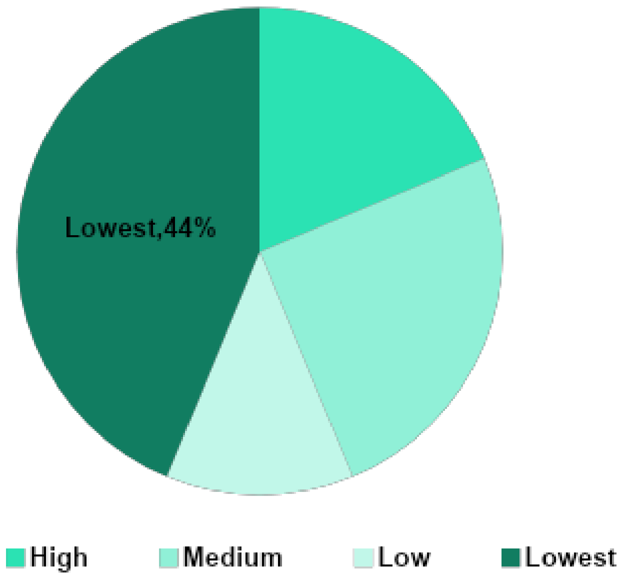
Hacken security consultants imitated the hacker activities to test the overall security state of the system.

The security review identified 3 high, 4 medium, 2 low and 7 lowest/best practice issues.

Most of medium and high-level vulnerabilities were already known by the Customer and for the moment work as expected.

According to the review auditors evaluate the security state of the system as moderate.

The distribution of Findings



Severity Definitions

Risk Level	Description
High	High-level vulnerabilities are easy in exploitation and may provide an attacker with full control of the affected systems, also may lead to significant data loss or downtime. There are exploits or PoC available in public access.
Medium	Medium-level vulnerabilities are much harder to exploit and may not provide the same access to affected systems. No exploits or PoCs available in public access. Exploitation provides only very limited access.
Low	Low-level vulnerabilities provide an attacker with information that may assist them in conducting subsequent attacks against target information systems or against other information systems, which belong to an organization. Exploitation is extremely difficult, or impact is minimal.

Lowest / Code Style / Best Practice	These vulnerabilities are informational and can be ignored.

Ambrosus Node and NOP Security Review

This section describes all performed actions against the target system. We outline task name, responsible, steps performed and findings with comments for each task.

0	Finalize Scope	Responsible	PR
Goal	Do decomposition of project, create detailed task list for the security review		
Steps Performed			
Consultant team had kick-off meeting with Customer engineers. Security engineers analyzed the potential threats for the system and formed a scope of the work			
Findings and Comments			
During the kick-off meeting auditors understood the main architectural concepts of the system - there are 4 main potential attacker classes: external attacker, external attacker with access to API, attacker that hosts Hermes node and attacker that hosts Atlas/Apollo node. Security engineers also obtained all necessary information to proceed with other tasks.			

1	Setup nodes	Responsible	SO

Goal	Setup 3 types of nodes (Apollo, Hermes and Atlas) for future testing
Steps Performed	
SO launched instances for each type of nodes; installed nodes with their dependencies; sent request to approve nodes' addresses.	
Findings and Comments	
Apollo, Atlas and Hermes nodes were deployed on Digital Ocean servers via NOP scripts. Parity client didn't sync with Ethereum network by default on each node. We needed to set "warp" parameter to true in parity_config.toml in order to start the nodes.	

2	Review of cryptoeconomics specification against potential threats	Responsible	PR/EM
Goal	Find high-level issues related to system architecture		
Steps Performed			
Consultants read the specification; analyzed what potential threats could be applied to the system; find obvious architecture issues			
Findings and Comments			

Cryptoeconomics specification is the document that describes system architecture. One of the main architectural concepts of the system is that main logic and verifications are handed down to smart contracts. Overall architecture security state is good, consultants found only 1 medium issue related to the specification.

3	Understanding the system by using its functionality	Responsible	Team
Goal	Gain deeper understanding of a system by security consultants		
Steps Performed			
Consultants followed the documentation and manually called different API functions of the node. Monitored the systems behavior via explorer, logs and proxies			
Findings and Comments			
No issues were discovered during manual test of the system. All the functions that were called by auditors worked as expected			

4	Permission checks against matrix	Responsible	EM
Goal	Confirm that permissions within the system are correctly implemented		
Steps Performed			

Auditors requested permission matrix and manually compared implementation logic against available documentation
Findings and Comments
Permissions are correctly implemented - all actions that require verification limit access as expected. Code implementation fully follows permissions matrix.

5	Calculate collision probability - analyze to write a report	Responsible	PR
Goal	Verify that the collision can't have significant impact on system		
Steps Performed			
Calculate the probability of collision for 1 billion of different entries. Analyze the impact of random id collision			
Findings and Comments			
IDs in the systems are hashes of the serialized json. We calculated the probability of collision for a billion of entries and it was less than 10^-10 %. We were informed by the customer that in case of collision for the assets and events Hermes node will just refuse to create second event/asset with identical id; in case of bundles, it just won't be uploaded to the network. It means that collision can't have serious impact on the system			

6	Autoscanning of the codebase	Responsible	EM
Goal	Check against typical security issues patterns with multiple security analysis tools		
Steps Performed			
Consultants launched and completed static code analysis using X, Y, Z, T applications security scanners. Auditors also run software composition analysis tools - npm audit and snyk			
Findings and Comments			
The outcome of static security scanners execution was: scanner X found 3 medium and 10 low issues; scanner Y found 1 critical and 2 medium issues; scanner Z found 7 medium issues; scanner T haven't found any issues; npm audit found 1 medium and 1 low issues; snyk found 1 high issue. These findings were manually reviewed during the task 16 of the project			

7	Analyze node upgradeability mechanism	Responsible	PR
Goal	Ensure that all nodes can securely update after important bugfixes		
Steps Performed			

Security engineers requested information about node upgradeability process. After that auditors analyzed the potential security issues of the process
Findings and Comments
"Each node owner is responsible for updating the nodes. When Customer releases the security update, it notifies node holders via emails gathered from KYC. After receiving notification node owner should manually update the node - login to node and run update.sh. Customer can check the current version of the node via node_info request. However, Customer don't have any integrity checks and the value can be abused. Docker don't setup latest containers so it won't update Parity or other in case of updates."

8	Review NOP script and analyze potential threats	Responsible	SO
Goal	Verify that NOP algorithm of generating config files are correct and the node is secure by default		
Steps Performed			
Consultants manually reviewed the source code of NOP; analyzed potential threats during setup process			
Findings and Comments			

<table>
<tr><td>NOP defines the default node configuration after set up. No potential attack vectors were discovered during review. However, there are no SSL advisory in NOP, default node will establish only HTTP connections without encryption.</td></tr>
</table>

<table>
<tr><td>9</td><td>Dump and analyze traffic between nodes</td><td>Responsible</td><td>EB</td></tr>
<tr><td>Goal</td><td colspan="3">Record and analyze traffic on three nodes (Apollo, Hermes, Atlas). It was necessary to understand the logic of each node and discovery the IP-addresses of other nodes. In the future, this information will be used for "DDoS" testing and "Network discovery + scanning of the nodes" testing</td></tr>
<tr><td colspan="4">Steps Performed</td></tr>
<tr><td colspan="4">Record traffic through “tcpdump”:
Analyze traffic through “Wireshark and NetworkMiner”
Nodes:
● Apollo - 139.59.208.7
● Hermes - 207.154.249.42
● Atlas - 46.101.137.241</td></tr>
<tr><td colspan="4">Findings and Comments</td></tr>
<tr><td colspan="4">Since all addresses of the nodes are known from the traffic analysis, the hacker can conduct a targeted attack on each of</td></tr>
</table>

the nodes separately. We recommend in the description on the launch of the node to make basic recommendations * Move nginx to another docker container * Make a white list for connection via ssh and set up a connection only by keys * Use only large(AWS, DO, etc) cloud providers.

10	Privilege escalation	Responsible	EB
Goal	Obtain high-level privileges (e.g. root privileges) and make their way to critical systems without being noticed (docker, nginx, source code, private key).		
Steps Performed			
When testing, we used several users with low privileges on the source system (docker, ubuntu, test). Source system: ● AWS machine image 'ambrosus-nop' ● DO pre-installed ubuntu 18.4 List of tests: Testing exploiting Kernel and Operating System Testing exploiting Applications and Services Testing exploiting Services which are running as root Testing exploiting SGID/SUID misconfiguration Testing exploiting sudo rights/user			

Testing exploiting badly configured cron jobs Testing exploiting Shell Escape Testing exploiting Symlinks Testing exploiting Buffer Overflow Testing exploiting Weak/reused/plaintext passwords Testing exploiting Bad path configuration
Findings and Comments
The tests did not show the presence of vulnerabilities, but we recommend setting up auto-update for all used services (kernel, ssh, nginx, docker, etc.)

11	Docker escape testing	Responsible	EB
Goal	Our goal was to escape from the container using the kernel or vulnerabilities in the docker itself to gain access to the node.		
Steps Performed			
● Testing all CVEs for docker (https://www.cvedetails.com/product/28125/Docker-Docker.html) ● Testing Kernel vulnerabilities ● Testing misconfiguration			
Findings and Comments			

Docker container escaped will generally use Docker Daemon file parsing vulnerabilities, system kernel privilege escalation vulnerabilities and other means, to achieve the purpose of elevating user rights and break the original isolation mechanism restrictions. According to its use of vulnerability points can be summarized as the use of Docker Daemon file parsing vulnerabilities to achieve the escape; the use of Docker container environment misconfigurations to achieve escape; use of kernel vulnerabilities to achieve escape three cases. Docker Daemon needs to compile the Dockerfile file, parsing image files, if the external input without filtering, when triggered to Docker Daemon loopholes, may cause container escaped. In the early version of the docker, compiling the deformed Dockerfile files and Improper parsing specially constructed soft link file in the images would cause arbitrary code execution, they all belong to this kind of escape problem. Kleindienst described in the article when mounted the /var/run/ directory to the container will lead to container escape, and if the CAP_DAC_READ_SEARCH privilege is given to the container by default can cause an arbitrary file access attack, they all belong to misconfiguration escape problem. Because the Docker container and the host share the same kernel, privilege escalation vulnerabilities in the Linux kernel and driver can be used to achieve container escape. Jian, Z in their paper

point out that can though be switching namespaces or through modifying shared memory achieve container escape. During testing was not found possible to escape from the container, but if you do not carry out regular updates of the docker and the image of the AWS 'ambrosus-nop' machine, this feature may appear

12	Fuzzing of APIs (all parameters in GET, POST, PUT requests)	Responsible	VS/DM
Goal	Find errors in the API. Bypass application logic. Accessing hidden data. Cause the node to stop working		
Steps Performed			
● Circumvent authentication and authorization mechanisms ● Escalate user privileges ● Hijack accounts belonging to other users ● Violate access controls placed by the administrator ● Alter data or data presentation ● Corrupt application and data integrity, functionality and performance ● Circumvent application business logic ● Circumvent application session management ● Break or analyze use of cryptography within user accessible components ● Sending requests with raw data ● Sending requests in the wrong format			
Findings and Comments			

During testing, no vulnerabilities were found in the API. There is one potential flaw that you can find in"Security Review Findings"

Method	REFERENCE	NoSQL injection	Fuzzing
POST	Create token	Protected	Protected
POST	Add account	Protected	Protected
GET	Find account	Protected	Protected
GET	Get account	Protected	Protected
PUT	Modify account	Protected	Protected
POST	Create an asset	Protected	Protected
GET	Fetch an asset by id	Protected	Protected
GET	Find assets	Protected	Protected
POST	Create an event	Protected	Protected
GET	Fetch event	Protected	Protected
GET	Find events	Protected	Protected
GET	Fetch bundle	Protected	Protected
GET	Fetch bundle metadata	Protected	Protected
GET	Get node info	Protected	Protected

13	NoSQL injection testing	Responsible	VS/DM
Goal	Gaining access to the database through NoSQL injection		

Steps Performed

All requests were checked in manual and automatic format for the presence of NoSQL injection.

Findings and Comments

During testing, no vulnerabilities were found in the API.

Method	REFERENCE	NoSQL injection	Fuzzing
POST	Create token	Protected	Protected
POST	Add account	Protected	Protected
GET	Find account	Protected	Protected
GET	Get account	Protected	Protected
PUT	Modify account	Protected	Protected
POST	Create an asset	Protected	Protected
GET	Fetch an asset by id	Protected	Protected
GET	Find assets	Protected	Protected
POST	Create an event	Protected	Protected
GET	Fetch event	Protected	Protected
GET	Find events	Protected	Protected
GET	Fetch bundle	Protected	Protected
GET	Fetch bundle metadata	Protected	Protected
GET	Get node info	Protected	Protected

14	Testing and code review of token generation	Responsible	PR
Goal	Verify that token is generated securely and attacker can't forge the token. Ensure that token authentication is secure		
Steps Performed			
Security engineers analyzed when token is used; manually reviewed the code responsible to token generation			
Findings and Comments			
Customer is aware and confirm that token should be used only for testing purposes and its usage is insecure by design. Customer don't recommend using the token in the mainnet. However, it is much more convenient for node holders to use the token and they can enable token authentication. Overall process of token generation is secure			

15	Manual review of timeout mechanism	Responsible	PR
Goal	Verify that default protection from DDoS and high-load is effective		
Steps Performed			
Auditors analyzed the mechanisms of timeouts for the node while receiving requests			

Findings and Comments
By default, Nodes don't have any application limits for requests; NOP don't recommend to implement any kind of DDoS protection, thus, there is no DDoS and high-load protection for the nodes. Timeout mechanism is implemented on nginx side on Customer servers. This mechanism was tested against DDoS during task 22 of the project.

16	Manual review of autoscanner findings	Responsible	EM
Goal	Discard all false positives from security scanners findings during stage 6 of the project		
Steps Performed			
Security consultants manually reviewed the findings of the autoscanners and tested their applicability			
Findings and Comments			
Scanners X, Y, Z, T together with npm audit and snyk found 22 different security issues. All of them were manually reviewed and none of them were valid.			

17	Analysis of KYC process	Responsible	PR
Goal	Verify that risk of malicious node set up is low		
Steps Performed			

Consultants obtained all information about the KYC process; analyzed the security risks of the process

Findings and Comments

"There are 2 different types of KYC processes for the node holders:

1. For Hermes node holders - the KYC process is light and most of the people can pass this KYC. It is done because Hermes node holders spend money in the system and difficult KYC process can push away potential Customer clients to deploy the node. As far as it is easy to pass Hermes KYC, Hermes node holder should be considered as attacker for other checks
2. For Atlas and Apollo node holders - the KYC process is more difficult. Firstly, KYC applicant should provide the proof of identity (for example, passport), secondly KYC applicant should provide proof of residence, lastly, Ambrosus does third-party background checks against applicant.

Note. KYC process for Atlas node is not currently implemented. Customer informed us that the process will be similar to Apollo node KYC

Considering all of the above, the risk of attacker being Apollo/Atlas node holder is low and Hermes endpoint might be used attacker for malicious activity"

18	Manual code review for immutability of data (merkle proofs etc.)	Responsible	EM
Goal	Verify the correctness of all tasks related to Ethereum blockchain. Confirm that the node correctly validates bundles, events and assets		
Steps Performed			
Auditors manually reviewed the implementation of bundles, events and assets validation; compared implementation logic against available documentation.			
Findings and Comments			
The node uses web3 package for all interaction with Ethereum blockchain. The implementation complies with best practices. The node uses ajv (https://www.npmjs.com/package/ajv) package to validate received data against JSON schemas. JSON schemas used in the system comply with documentation. No issues related to the data immutability were found			

19	Analyze potential deserialization vulnerabilities	Responsible	PR
Goal	Verify that serialization and deserialization is done in secure way		
Steps Performed			

Auditors analyzed how and what data types are serialized and deserialized. Manually reviewed the code of the object serialization, particularly against https://www.acunetix.com/blog/web-security-zone/deserialization-vulnerabilities-attacking-deserialization-in-js/
Findings and Comments
System mostly uses serialization to store the json in the database or calculate a hash of the data. JSON.stringify and JSON.parse are used for json serialization and deserializations that is considered to be secure. Moreover, all json data is validated against predefined schema. Serialization for objects is used only in serializeForHashing function, however the objects passed to the function are never deserialized. It means that code injection via deserialization can't be performed.

20	Web pentest for Hermes client side	Responsible	VS/DM
Goal	Search for errors and vulnerabilities in web applications such as xss, sqli, ssti, csrf, idor etc.		
Steps Performed			
Client is requesting Consultant assistance in the performance of grey-box web application security assessment that will include the following components: ● Architecture security review ● Web applications described in the scope			

- Mapping application code against industry best practices OWASP ASVS (https://goo.gl/NB9NT6)

The stated objectives of this assessment are:

- Circumvent authentication and authorization mechanisms
- Escalate user privileges
- Hijack accounts belonging to other users
- Violate access controls placed by the administrator
- Alter data or data presentation
- Corrupt application and data integrity, functionality and performance
- Circumvent application business logic
- Circumvent application session management
- Break or analyze use of cryptography within user accessible components

Application will be verified for common vulnerabilities such as the OWASP Top 10, logical mistake of application work.

Findings and Comments

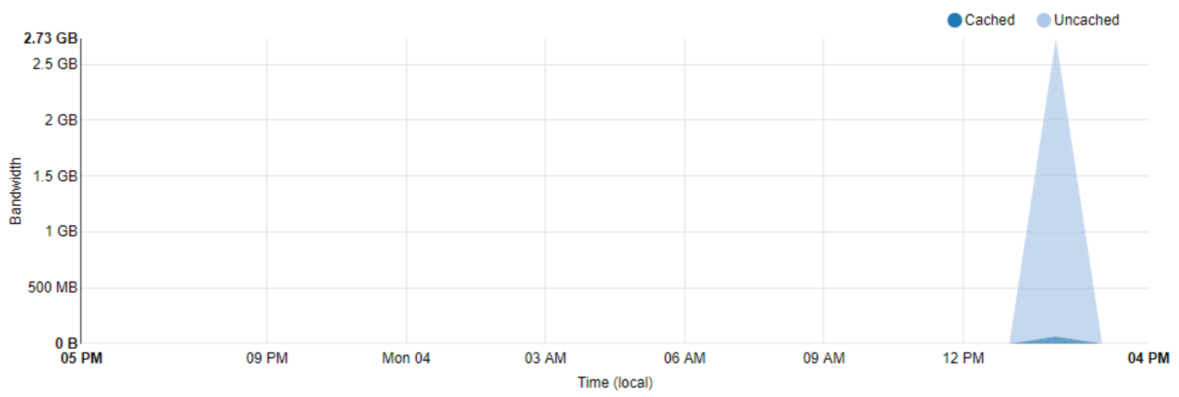
For all sites, we recommend connecting WAF and DDoS protection or using a professional/corporate plan in the CloudFlare. You can look at all found defects in "Security Review Findings"

21	Network discovery + scanning of the nodes	Responsible	EB
Goal	Identify active hosts and services, for up to the total number of in-scope active IP addresses (Main nodes and user nodes. We received this list while recording and analyzing traffic.), and assess a security posture of those systems. Attempt to exploit identified		

	vulnerabilities and demonstrate the impact of those vulnerabilities.
Steps Performed	
Getting a list of active nodes and scanning for vulnerabilities	
Findings and Comments	
During testing, no vulnerabilities were found in the external network.	

22	DDoS simulation	Responsible	EB
Goal	Check the operation of the cloud provider, check the system response to DDoS (HTTP, TCP, UDP), find flaws in the operation of the system and its response to DDoS		
Steps Performed			
Test cases (Common DDoS attack vectors (L3, L4 & L7)): 1. HTTP get flood 2. SYN flood 3. HTTP slowloris 4. ICMP flood The speed of load testing varies from 100 MB/s to 6-7 GB/s			
Findings and Comments			
With a DNS flood, the server crashed in a few minutes, but aws quickly blocked malicious traffic			

Host	Type	0-1000 MB/s	2-10 GB/s	10-50 GB/s
34.247.98.162	UDP Flood	PASS	PASS	PASS
13.126.51.11	UDP Flood	PASS	PASS	PASS
52.215.227.185	UDP Flood	PASS	PASS	PASS
34.247.98.162	TCP Flood	PASS	PASS	PASS
13.126.51.11	TCP Flood	PASS	PASS	PASS
52.215.227.185	TCP Flood	PASS	PASS	PASS
34.247.98.162	DNS Flood	PASS	PASS	PASS
13.126.51.11	DNS Flood	PASS	PASS	PASS
52.215.227.185	DNS Flood	PASS	PASS	PASS
34.247.98.162	30303 Flood	PASS	PASS	PASS
13.126.51.11	30303 Flood	PASS	PASS	PASS
52.215.227.185	30303 Flood	PASS	PASS	PASS

23	Analyze private key storage and usage	Responsible	PR/EM
Goal	Verify that private key is stored securely and attacker can't get a private key if he gets access to the host		
Steps Performed			
Consultants searched for the private key on the host; analyzed where private key is stored or used in the codebase			
Findings and Comments			
Private key is used for all signatures - to sign assets, events and bundles. This is the only functionality that uses a private key. Private key is stored in clear text on the node in docker-compose.yaml and state.json files. During code review auditors also discovered that private key can be written to the logs in some conditions.			

24	Analysis of cryptography implementation	Responsible	EM
Goal	Verify that all cryptography used is implemented/used correctly		
Steps Performed			
Auditors searched for crypto primitives in the codebase and dependencies			
Findings and Comments			
The only crypto primitives used within the system are keccak256 for hashing and ECDSA for signing and verifying signatures. These functions are implemented in web3 library, no custom cryptography is used within the project. Considering abovementioned the cryptography implementation is secure.			

25	Report development	Responsible	EB/PR
Goal	Prepare final report that will be presented to Customer		
Steps Performed			
Assemble description of all steps performed by the team and corresponding findings			
Findings and Comments			
N/A			

Security Review Findings

The section contains all security and best practice findings found during security review with their severities, impact and mitigation recommendations.

1	Private key is logged	Severity	High
Description			
Private key is logged via `logger.info('Secret: ${account.secret}');` during node initialization.			
Impact			
It might be easier for the attacker to stole private key from the logs than from the node itself			
How to mitigate			
Don't log private key anyway. It is recommended not to work directly with the private key.			
Corresponding task in security review			23

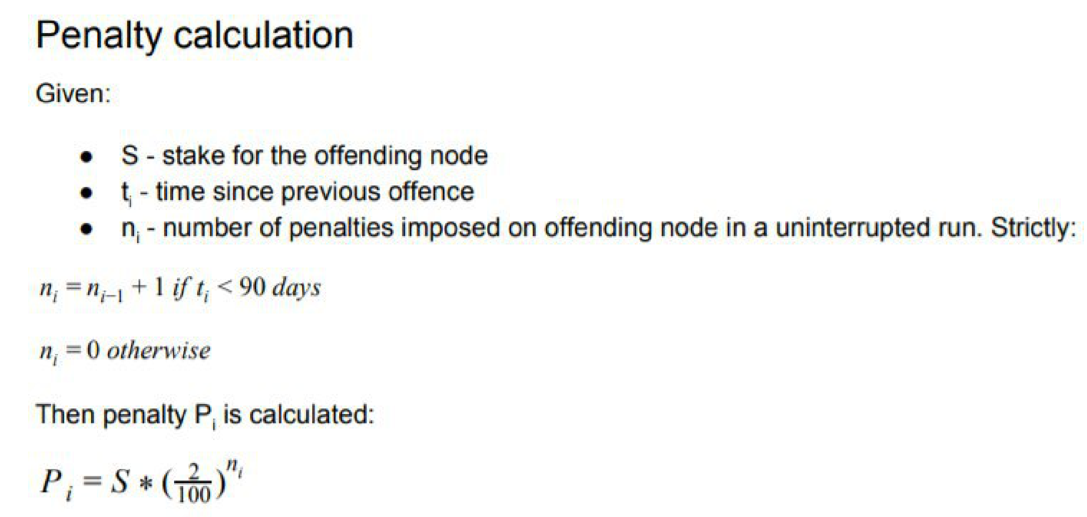
2	Private key and passwords for unlocking accounts are stored as plain text on the node	Severity	High
Description			
Private key in docker-compose.yaml and state.json files and password for unlocking accounts (signer, private account, validators) are stored as plain text on the node.			
Impact			

If attacker gets access to the node - he gets access to the Ethereum private key. He can withdraw all the founds on the account using it	
How to mitigate	
We recommend using signer middleware for the system. It can be deployed in separate container and contain a private key that never leaves the signer. Node can request transactions sign from the signer to validate bundles. The signer should have transaction filter that whitelists only necessary transactions, for example, to sign a bundle. If attacker gets access to the node, he could only execute whitelisted transaction and he can't transfer funds from it. In order to obtain a private key, he will need to get access to the signer, where private key is stored. Clef (https://github.com/ethereum/go-ethereum/tree/master/cmd/clef) is an example of signer implementation. Clef's security can be used for the system.	
Corresponding task in security review	23

3	Yoast SEO Authenticated Race Condition	Severity	High
Description			
Current Yoast version has a race condition vulnerability which leads to command execution. The command executions can be			

exploited with any SEO Manager role account. The detailed description of vulnerability can be found here - https://thattechguy.com.au/yoast-seo-authenticated-race-condition/ Vulnerable endpoint https://tech.ambrosus.com/	
Impact	
Vulnerability allows you to elevate your privileges on the server and execute commands from a privileged user.	
How to mitigate	
Consider upgrading Yoast SEO to the latest version	
Corresponding task in security review	20

4	Penalty calculation issue	Severity	Medium
Description			
The formula for Penalty calculation is the following.			

 Penalty calculation Given: - S - stake for the offending node - t_i - time since previous offence - n_i - number of penalties imposed on offending node in a uninterrupted run. Strictly: $n_i = n_{i-1} + 1 \ if \ t_i < 90 \ days$ $n_i = 0 \ otherwise$ Then penalty P_i is calculated: $P_i = S * (\frac{2}{100})^{n_i}$ Thus, the penalty withdrawn exponentially decreases with the number of punishments. Since \sum_{i=1}^{\infty}(2/100)^i = 1/49 \approx 0.0204, the offending node will be fined in total for all times not more than 3% of the stake	
Impact	
1) The more the node will be punished, the less motivation will have the others to challenge it. 2) The reward for sheltering is given to the node continuously. Thus, the node that was punished several times will still be able to profit even in case of challenges	
How to mitigate	
Consider reviewing the penalty formula making the penalty exponentially increasing instead of decreasing	
Corresponding task in security review	2

5	MongoDB access control is not implemented	Severity	Medium
Description			
Organization with Hermes node might have read access to other organization bundles. The issues are known and confirmed by Customer. Customer already works on the fix.			
Impact			
Attacker who setup malicious Hermes node might have read access to all bundles within the system			
How to mitigate			
Implement access control for Hermes nodes for MongoDB - Hermes node should have access to their local database and don't have access to all bundles			
Corresponding task in security review			3

6	Docker images for parity and mongo doesn't use latest images	Severity	Medium
Description			
NOP configures docker-compose.yml with non-latest version of docker images for parity and mongo (parity/parity:v2.0.8 and mongo:4.1)			
Impact			
Old versions of the docker images potentially contain unfixed bugs and vulnerabilities			
How to mitigate			

Change all versions to latest in the NOP	
Corresponding task in security review	8

7	WordPress XML-RPC authentication brute force	Severity	Medium
Description			
The XML-RPC API that WordPress provides gives developers a way to write applications (for Customer) that can do many of the things that you can do when logged into WordPress via the web interface. The main weaknesses associated with XML-RPC are: Brute force attacks: Attackers try to login to WordPress using xmlrpc.php Vulnerable endpoint https://tech.ambrosus.com/xmlrpc.php			
Impact			
A hacker can find the right combination login / password combination for https://tech.ambrosus.com/ and access the server			
How to mitigate			
It is necessary to disable the XML-RPC on https://tech.ambrosus.com/			
Corresponding task in security review			20

8	Synchronization fails with warp == false	Severity	Low

Description	
NOP generates parity_config.toml with warp == false by default. It makes synchronization unavailable.	
Impact	
It is not easy to understand where is problem and potentially could lead to bigger issues during fixing process.	
How to mitigate	
Set warp == true for synchronization.	
Corresponding task in security review	1

9	No SSL configuration in NOP	Severity	Low
Description			
After NOP configuration nodes accept http by default.			
Impact			
Default configuration of a masternode makes man-in-the-middle attack possible.			
How to mitigate			
Accept only https requests, add https configuration to the NOP			
Corresponding task in security review			8

10	Outdated nodes prices	Severity	Lowest
Description			
Node KYC page contains outdated prices (https://tech.ambrosus.com/apply/). For example, Hermes node			

setup is free of charge, however, application page says that node holder should pay 150k AMB for it	
Impact	
It misleads AMB masternode holders and potentially increase their spending.	
How to mitigate	
Update KYC page	
Corresponding task in security review	2

11	Token access functionality should be removed from the repository	Severity	Lowest
Description			
As far as, token functionality is already developed, node holders can allow token access for better usability. Token is stored in HTTP header and can be stolen via different attacks			
Impact			
Node holders can potentially enable insecure token authentication functionality			
How to mitigate			
Remove token authentication from the codebase			
Corresponding task in security review			14

12	Potential reflected XSS	Severity	Lowest
Description			
There is no escaping of special characters on the server.			

http://207.154.249.42/assets/0x826c18a159ff481f5383984e3cca525d78e6a40450564e683baa0cf616be24c4'"><img src="1" onerror=":alert(1)">	
Impact	
The issue doesn’t have proven security impact, however, it is recommended to validate GET parameters	
How to mitigate	
You need to add shielding of characters or connect the WAF to block all malicious traffic (Allows protection even from theoretical attacks)	
Corresponding task in security review	12

13	Missing Security Headers	Severity	Lowest
Description			
This defect is present on all sites https://*.ambrosus.com HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value "Strict-Transport-Security: max-age=31536000; includeSubDomains". Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of			

approved content, you can prevent the browser from loading malicious assets.

X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "X-Frame-Options: SAMEORIGIN".

X-XSS-Protection sets the configuration for the cross-site scripting filter built into most browsers. Recommended value "X-XSS-Protection: 1; mode=block".

X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".

Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.

Feature Policy is a new header that allows a site to control which features and APIs can be used in the browser.

Impact

The absence of these headers makes the server less secure and it cannot block attacks like XSS or Clickjacking.	
How to mitigate	
Add additional security headers to all servers that are listed above	
Corresponding task in security review	20

14	Account bruteforce / Username enumeration / Email spamming	Severity	Lowest
Description			
Due to the lack of a captcha or other protection mechanism on the site https://dashboard.hermes.ambrosus-test.com/, a hacker can execute requests without restrictions and blocking. POST request to https://hermes.ambrosus-test.com/extended/account/secret allows you to hack (brute force) an account and determine whether a user is registered or not POST request to https://hermes.ambrosus-test.com/extended/organization/request allows you to register new accounts on any mail (allows you to blacklist your email server) and determine whether a user is registered or not			
Impact			

Attacker might brute force access to the accounts; might block the mail server	
How to mitigate	
Add a captcha or other protection mechanism (WAF or one-time token). We recommend connecting hidden Google Captcha (https://www.google.com/recaptcha/intro/v3.html) to all functional queries or connect CloudFlare for all subdomains and set the rate limit for the necessary pages.	
Corresponding task in security review	20

15	No integrity checks for deployed nodes	Severity	Lowest
Description			
There are no integrity checks mechanism within the system. Node holders might change the codebase before deployment. Security mechanisms that are preventing from this is KYC and crucial verifications on smart contract layer.			
Impact			
It is known and desired behavior of the system, however, it makes much bigger attack surface for the attacker			
How to mitigate			
Consider disallowing node code changes before the node deployment			

Corresponding task in security review	7

16	Usage of non-latest versions of libraries	Severity	Lowest
Description			
The version of web3 used in the system is 1.0.0-beta.34, however, the latest is 1.0.0-beta.38 as for now; the version of ajv used in the system 6.5.5, however, the latest is 6.7.0 as for now			
Impact			
Issues doesn't have security impact, represents best practice recommendation			
How to mitigate			
Update the libraries listed above			
Corresponding task in security review			18

Conclusion

Node code was manually reviewed and analyzed with static analysis tools.

NOP scripts were manually reviewed, and risk assessment was performed for it.

The system’s network was tested via fuzzing and DDoS.

All web endpoints were tested against typical web vulnerabilities.

This document describes methodology, and all performed actions in Ambrosus Node and NOP Security Review section.

Security review report contains all found security vulnerabilities and other issues in the reviewed code.

Overall quality of reviewed code is high; however, the security state is moderate containing 3 high and 4 medium severity vulnerabilities.

Disclaimers

Hacken Disclaimer

The smart codebase given for review have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the source code, the details of which are disclosed in this report, web part vulnerabilities, deployment and functionality (performing the intended functions).

The review makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the system. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of the system.