



QuillAudits

# Audit Report April, 2024

For



# Table of Content

Executive Summary .....	02
Number of Security Issues per Severity .....	03
Checked Vulnerabilities .....	04
Techniques and Methods .....	05
Types of Severity .....	07
Types of Issues .....	07
<b>High Severity Issues</b>	08
Issues In the Production Network Configuration	08
1. LDAP	08
2. Cryptogen ( certificates are generated using cryptogen )	09
3. Enrolling and registering Users API changes	10
4. Missing Root CA	11
<b>Medium Severity Issues</b>	12
5. No. of CAs	12
<b>Low Severity Issues</b>	13
<b>Informational Issues</b>	13
Closing Summary .....	14
Disclaimer .....	14



# Executive Summary

## Project Name

Agira Tech

## Timeline

15th April 2024 - 29th April 2024

## Audit Scope

The scope of this audit was to analyse the AgiraTech for quality, security, and correctness.

## Source Code

[https://agiratechnologiesinc-my.sharepoint.com/:f:/g/personal/praveen\\_m\\_agiratech\\_com/Er1dqcfHI2RBj9bSLcalQPIBrD1QxmzVTHLZUMflgU74zQ?e=GxygjK](https://agiratechnologiesinc-my.sharepoint.com/:f:/g/personal/praveen_m_agiratech_com/Er1dqcfHI2RBj9bSLcalQPIBrD1QxmzVTHLZUMflgU74zQ?e=GxygjK)



# Number of Security Issues per Severity



High

Medium

Low

Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	4	1	0	0

# Checked Vulnerabilities

- ✓ Denial of Service (DoS) Attacks
- ✓ Consensus Manipulation
- ✓ MSP Compromise
- ✓ Smart Contract Exploitation
- ✓ Chaincode Injection
- ✓ Insecure Key Management
- ✓ Misconfigured Endorsement Policies
- ✓ Lack of Secure Communication
- ✓ Insufficient Logging and Monitoring
- ✓ Identity Spoofing
- ✓ Chaincode Bugs
- ✓ Data Leakage in Private Data Collections
- ✓ Lack of Regular Security Updates
- ✓ Insider Threats
- ✓ Adding of docprom
- ✓ Missing of LDAP
- ✓ Missing Root CA
- ✓ ICA's not created using Root CA certs
- ✓ Use of Cryptogen instead of LDAP
- ✓ MSP policies not configured properly
- ✓ Enrolling and registering Users API changes





# Techniques and Methods

Throughout the Audit, care has been taken for

## Network Configuration Review:

- **In-depth examination:** A thorough analysis of the network configuration is conducted, focusing on channels, organisations, and peers. This ensures a clear understanding of the network structure and its participants.
- **Access control verification:** The audit meticulously verifies that access control policies are accurately defined and enforced. This safeguards the network against unauthorised access attempts.
- **Cryptographic material security assessment:** The secure management of cryptographic materials (certificates and keys) is critically evaluated. This ensures the confidentiality and integrity of transactions within the network.

## Chain Code Review:

- **Vulnerability analysis:** The smart contract code (chaincode) undergoes a rigorous examination for vulnerabilities. This includes identifying and mitigating common security flaws like improper input validation, resource exhaustion, and logic errors.
- **Business logic scrutiny:** The audit pays close attention to potential business logic errors in smart contracts that might lead to unintended consequences or financial losses.

## Access Control and Identity Management:

- **Membership Service Provider (MSP) evaluation:** The configuration of the MSP, which governs the issuance and management of identities within the network, is meticulously reviewed.
- **Identity management validation:** The audit verifies that identities are issued, managed, and revoked according to established protocols to prevent unauthorised access.
- **Unauthorized access detection:** The process actively searches for any potential unauthorised access paths or loopholes that could be exploited by malicious actors.

# Techniques and Methods

## Consensus Mechanism and Orderer Security:

- **Consensus algorithm understanding:** The audit delves into the specific consensus algorithm employed by the network (e.g., Crash Fault Tolerant or Byzantine Fault Tolerant) to comprehend its security posture.
- **Orderer security assessment:** The security configuration and operational integrity of the orderer nodes are meticulously evaluated to ensure they operate as intended.
- **Consensus activity monitoring:** The audit establishes mechanisms to monitor the consensus process for any suspicious activity that might indicate tampering or manipulation.

## Channel Security:

- **Channel policy and permission evaluation:** The audit thoroughly examines the channel policies and permissions to guarantee that only authorized peers can participate in specific channels.
- **Channel misconfiguration detection:** The process actively identifies any potential misconfigurations in channel settings that could compromise network security.

## Data Privacy and Confidentiality:

- **Data privacy feature assessment:** The audit evaluates the network's data privacy features, such as private data collections, to understand how sensitive data is protected.
- **Sensitive data protection verification:** The audit verifies that sensitive data is encrypted or otherwise protected throughout its lifecycle to prevent unauthorized access or disclosure.

## Logging and Monitoring:

- **Logging mechanism review:** The audit assesses the existing logging mechanisms within the network to ensure comprehensive activity capture for security analysis.
- **Security incident monitoring:** The process establishes procedures to monitor for abnormal behavior or security incidents that could indicate a breach or attack.
- **Performance analysis consideration:** The audit explores the potential of tools like Hyperledger Caliper for performance analysis, which can indirectly contribute to identifying potential security bottlenecks.



## Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

### High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

### Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

### Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

### Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

## Types of Issues

### Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

### Resolved

These are the issues identified in the initial audit and have been successfully fixed.

### Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

### Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.





## 1. LDAP

### Issues In the Production Network configuration

The Fabric CA server can be configured to read from an LDAP server. In particular, the Fabric CA server may connect to an LDAP server to do the following: authenticate an identity prior to enrollment retrieve an identity's attribute values which are used for authorization.

<https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html#configuring-ldap>

### Benefits of LDAP Integration:

1. **Improved Security:** LDAP allows for centralized user management and authentication, enhancing security by leveraging existing user credentials. This eliminates the need to store user credentials within the Fabric CA server itself.
2. **Scalability:** When dealing with a large number of users and entities, LDAP simplifies enrollment by offloading user management from the Fabric CA server. This improves scalability for managing user identities.

### Important Considerations:

1. **Reduced CA Load:** While LDAP can minimize direct requests to the CA server for authentication, certificate issuance still happens through the CA. Requests for enrollment and certificate renewal will still go to the CA server.
2. **Network Efficiency:** The overall network efficiency might improve by offloading user authentication to LDAP, but it depends on the network structure and LDAP server location.
3. **Complexity:** Implementing and maintaining LDAP adds complexity to the system.

### Here's a more nuanced view of your recommendation:

Enabling LDAP is a good security practice for managing user identities when dealing with a significant number of users, especially if you already have an existing LDAP infrastructure. However, it won't entirely eliminate requests to the CA server.

1. **Security of the LDAP Server:** Ensure the LDAP server is secure to prevent unauthorized access and potential compromise of user credentials.  
**Performance:** Evaluate the performance impact of LDAP integration on your network, especially if the LDAP server is not located within the same physical location as the Fabric CA server.

### Additional Factors to Consider:

1. **Security of the LDAP Server:** Ensure the LDAP server is secure to prevent unauthorized access and potential compromise of user credentials.
2. **Performance:** Evaluate the performance impact of LDAP integration on your network, especially if the LDAP server is not located within the same physical location as the Fabric CA server.

### Conclusion

LDAP integration offers a secure and scalable approach to managing user identities for Fabric CA, particularly for large user bases.

### Status

**Resolved**

## 2. Cryptogen ( certificates are generated using cryptogen )

### Issues In the Production Network configuration

**Lack of Revocation:** cryptogen doesn't offer functionalities for certificate revocation. In a production environment, the ability to revoke compromised certificates is crucial for maintaining network security.

### Management and Control Issues:

- **Limited Functionality:** cryptogen generates a basic set of cryptographic materials and lacks the advanced features of Fabric CA. It doesn't support granular control over certificate lifespans, enrollment permissions, or user attributes.
- **Decentralized Management:** Materials generated with cryptogen are stored locally, making it difficult to manage and revoke certificates in a production setting. A central authority is essential for managing identities and certificates at scale.
- **Reproducibility Challenges:** While cryptogen offers some level of consistency, it might not be ideal for replicating complex production environments where precise control over material generation is necessary.

**The use of libraries like openssl is recommended for more scalability and features.**

<https://hyperledger-fabric.readthedocs.io/en/latest/commands/cryptogen.html>

### Status

**Resolved**



#### Description

When transitioning a REST API from using a Fabric CA server to LDAP for user enrollment and authentication, the following changes and considerations are typically involved:

1. **Authentication Mechanism:** Instead of authenticating users against the Fabric CA server's user registry, the REST API will now authenticate users against the LDAP server. This involves updating the authentication logic within the REST API to use LDAP protocols (like LDAP bind operations) for verifying user credentials.
2. **User Enrollment:** Previously, user enrollment while registering users directly with the Fabric CA server using its APIs. With LDAP, user enrollment refers to creating user entries within the LDAP directory itself. This include attributes such as username, password, email, roles, and other relevant metadata.
3. **Integration Setup:** The REST API needs to establish connectivity and integration with the LDAP server. This includes configuring the REST API to communicate over LDAP protocols (typically LDAP or LDAPS - LDAP over SSL/TLS). Integration details such as LDAP server hostname, port, bind DN (Distinguished Name), and credentials (if required for binding) must be configured securely within the REST API environment.
4. **Handling User Roles and Attributes:** LDAP directories organize users into hierarchical structures and allow for the definition of attributes and group memberships. The REST API needs to adapt its logic to retrieve user roles and attributes from LDAP entries when performing access control and authorization checks.
5. **Security Considerations:** LDAP communications should be secured using appropriate protocols (e.g., LDAPS) to protect user credentials and data during transmission. Additionally, access controls within the LDAP directory should be configured to restrict unauthorized access to user information.
6. **Error Handling and Logging:** Error handling mechanisms within the REST API should be updated to appropriately manage LDAP-specific errors and exceptions. Logging mechanisms may also need adjustment to capture relevant LDAP-related events for monitoring and troubleshooting purposes.

## Remediation

By transitioning from Fabric CA server to LDAP for user enrollment and authentication, the REST API achieves integration with a centralized identity management system, enhances scalability, aligns with enterprise-wide authentication standards, and ensures consistent user management practices across different applications and services within the organization. This transition also simplifies the management of user identities by leveraging LDAP's robust directory services capabilities.

## Status

**Resolved**

## 4. Missing Root CA

### Description

In a robust network setup, the presence of a Root Certificate Authority (CA) is crucial. Typically, each organization should ideally have its own Root CA, from which Intermediate CAs (ICAs) can be generated. These ICAs then facilitate the issuance of certificates for various entities within the organization, such as peers and orderers in a Hyperledger Fabric network. This hierarchical structure not only enhances security but also offers flexibility in managing certificates and ensuring trust across the network.

When it comes to Hyperledger Fabric, the Fabric CA (Certificate Authority) server plays a pivotal role in managing identities and certificates within the network. It enables organizations to establish a private CA infrastructure tailored to their specific needs. The Fabric CA server supports the creation and management of multiple CAs, including intermediate CAs, aligning with the best practices for security and flexibility in certificate management.

### Remediation

By implementing a Root CA and leveraging Fabric CA servers to manage ICAs, organizations can maintain a structured approach to certificate issuance and validation. This setup not only enhances security but also simplifies the administration of certificates across different organizational units within the Hyperledger Fabric network.

Therefore, ensuring the presence of a Root CA, supported by Fabric CA servers for managing ICAs, is essential for establishing a secure and flexible certificate management framework in Hyperledger Fabric networks.

## Status

**Resolved**





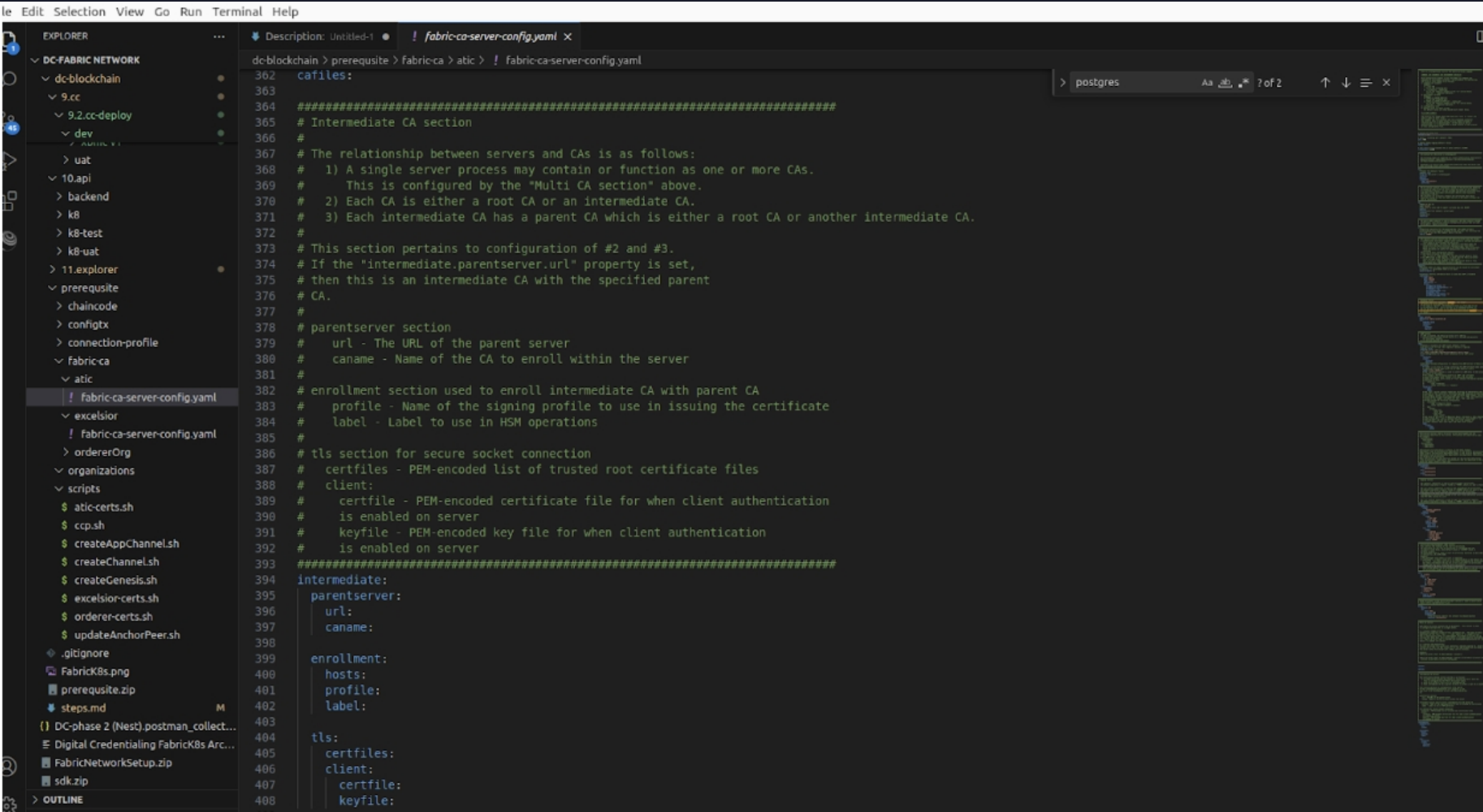
# Medium Severity Issues

## 5. No. of CAs

### Description

In a production network, it is recommended to deploy at least one CA per organization for enrollment purposes and another for TLS. For example, if you deploy three peers that are associated with one organization and an ordering node that is associated with an ordering organization, you will need at least four CAs. Two of the CAs will be for the peer organization (generating the enrollment and TLS certificates for the peer, admins, communications, and the folder structure of the MSP representing the organization) and the other two will be for the orderer organization. Note that users will generally only register and enroll with the enrollment CA, while nodes will register and enroll with both the enrollment CA (where the node will get its signing certificates that identify it when it attempts to sign its actions) and with the TLS CA (where it will get the TLS certificates it uses to authenticate its communications).

In your case There should be one Root CA for the whole network then using root CA to create multiple ICAs ( for each organization ) . This is the recommended approach for better scalability and will be easy whenever you want to revoke certificates of the whole organization if it demands.



```
362 cafiles:
363
364 #####
365 # Intermediate CA section
366 #
367 # The relationship between servers and CAs is as follows:
368 # 1) A single server process may contain or function as one or more CAs.
369 # This is configured by the "Multi CA section" above.
370 # 2) Each CA is either a root CA or an intermediate CA.
371 # 3) Each intermediate CA has a parent CA which is either a root CA or another intermediate CA.
372 #
373 # This section pertains to configuration of #2 and #3.
374 # If the "intermediate.parentserver.url" property is set,
375 # then this is an intermediate CA with the specified parent
376 # CA.
377 #
378 # parentserver section
379 # url - The URL of the parent server
380 # caname - Name of the CA to enroll within the server
381 #
382 # enrollment section used to enroll intermediate CA with parent CA
383 # profile - Name of the signing profile to use in issuing the certificate
384 # label - Label to use in HSM operations
385 #
386 # tls section for secure socket connection
387 # certfiles - PEM-encoded list of trusted root certificate files
388 # client:
389 #   certfile - PEM-encoded certificate file for when client authentication
390 #   is enabled on server
391 #   keyfile - PEM-encoded key file for when client authentication
392 #   is enabled on server
393 #####
394 intermediate:
395   parentserver:
396     url:
397     caname:
398
399   enrollment:
400     hosts:
401     profile:
402     label:
403
404   tls:
405     certfiles:
406     client:
407       certfile:
408       keyfile:
```



### Reference link in Hyperledger fabric doc

[https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment\\_guide\\_overview.html#dg-step-two-set-up-a-cluster-for-your-resources](https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment_guide_overview.html#dg-step-two-set-up-a-cluster-for-your-resources)

### Status

Resolved

## Informational Issues

No issues were found.

## Low Severity Issues

No issues were found.



# Closing Summary

In this report, we have considered the security of the AgiraTech codebase. We performed our audit according to the procedure described above.

Some issues of high and medium severity were found, and some suggestions and best practices were also provided in order to improve the code quality and security posture. In The End, the AgiraTech Team Resolved all Issues.

## Disclaimer

QuillAudits security audit provides services to help identify and mitigate potential security risks in AgiraTech. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of AgiraTech. One audit is enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the AgiraTech to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.



# About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over \$30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.



**1000+**

Audits Completed



**\$30B**

Secured



**1M+**

Lines of Code Audited



## Follow Our Journey



# Audit Report April, 2024

For



QuillAudits

📍 Canada, India, Singapore, UAE, UK

🌐 [www.quillaudits.com](http://www.quillaudits.com)

✉️ [audits@quillhash.com](mailto:audits@quillhash.com)