Cyberscope

A TAC Security Company

Audit Report

MCN Chain

October 2025

Repository https://github.com/mcnchain/mcnchain

Commit 1298b61ad9e5d3bc3187459de0e406a46c90b715

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Readability Comment

The audit scope is to check for security vulnerabilities, validate the business logic, and propose potential optimizations. The codebase is highly fragmented, deviates from standard security principles, and references non-existent structures. Even if all identified issues were addressed, the project remains non-production-ready. The development team is strongly advised to consolidate the code, thoroughly test its functionality and installation process, and restructure to improve readability, maintainability, and overall consistency.

Review

Repository	https://github.com/mcnchain/mcnchain
Commit	1298b61ad9e5d3bc3187459de0e406a46c90b715

Audit Updates

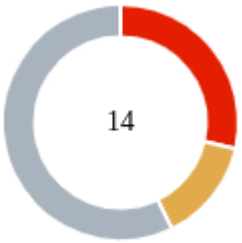
Initial Audit	29 Oct 2025

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Critical	EPC	Empty Password Configuration	Unresolved
Critical	IAC	Inconsistent Address Configuration	Unresolved
Critical	IPL	Invalid Path Lookup	Unresolved
Critical	IRQ	Invalid Regex Query	Unresolved
Medium	EBK	Exposed Bootnode Key	Unresolved
Medium	ECD	Exposed Coinbase Data	Unresolved
Minor / Informative	DPC	Deprecated Peer Configuration	Unresolved
Minor / Informative	EBO	Existent Bootnode Overwrite	Unresolved
Minor / Informative	ECO	Existent Coinbase Overwrite	Unresolved
Minor / Informative	HV	Hardcoded Values	Unresolved
Minor / Informative	NPO	Non-Persistent Operations	Unresolved
Minor / Informative	PNI	Public Node Interface	Unresolved
Minor / Informative	USV	Unused State Variables	Unresolved
Minor / Informative	ZGT	Zero Genesis Timestamp	Unresolved

Findings Breakdown



Critical	4
Medium	2
Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	4	0	0	0
Medium	2	0	0	0
Minor / Informative	8	0	0	0

EPC - Empty Password Configuration

Criticality	Critical
Location	01_make_accounts.sh
Status	Unresolved

Description

The script generates a new Ethereum account using Geth and references a password file for encryption:

```shell
ADDR_RAW="$(geth account new --datadir "$DATADIR"
--password "$PASSWORD_FILE" 2>&1 | tee /dev/stderr)"
```

However, it does not verify whether the password file contains a non-empty password. If the file is empty, Geth will proceed to create the account with no password protection.

An empty password leaves the private key unprotected, allowing anyone with access to the keystore file to unlock the account. This significantly compromises the security of the validator and could result in financial loss.

Recommendation

It is recommended to validate that the password file contains a non-empty password before creating the account. Using a strong, high-entropy password enhances account security by making it more resistant to brute-force and dictionary attacks.

IAC - Inconsistent Address Configuration

Criticality	Critical
Location	03_render_static_nodes.sh
Status	Unresolved

Description

The network setup script retrieves validator addresses and the bootnode ENODE from the following files:

```shell
VAL1=$(cat node/validator1/address.txt)
VAL2=$(cat node/validator2/address.txt)

ENODE=$(cat node/bootnode/enode.txt)
```

However, the script does not verify whether these files exist, are non-empty, or contain valid Ethereum addresses. This lack of validation can lead to runtime errors or misconfigurations.

Furthermore, the account creation script (`01_make_accounts.sh`) only generates a `.coinbase` file for a validator account and does not create the required `address.txt` files. As a result, the setup script will fail unless these files are manually created.

In the current configuration, only `validator1` is included in the initiator script, meaning `validator2` 's address file is missing by default. This introduces a manual dependency, reducing the reproducibility of the setup process and increasing the probability of misconfiguration.

Recommendation

It is recommended to validate that all required address files exist and are non-empty before proceeding with the network setup. These files should also be checked to ensure they contain valid Ethereum addresses. To improve reproducibility and reduce manual intervention, the account creation process or the network setup script should be updated to automatically generate the required address.txt files from the coinbase for each validator. This ensures that all validators are properly configured without relying on manually created files.

IPL - Invalid Path Lookup

Criticality	Critical
Location	06_run_validator1.sh 07_run_validator2.sh
Status	Unresolved

Description

The `06_run_validator1` and `07_run_validator2` scripts reference a `./genesis/mainnet.genesis.json` file that does not exist and is not generated by any prior step in the execution sequence. This missing file causes an inconsistency that prevents the validator nodes from initializing correctly.

```
Shell
GENESIS="${GENESIS:-./genesis/mainnet.genesis.json
}"
```

Recommendation

The team must ensure the network configuration file is queried correctly to guarantee proper initialization. Accurate retrieval of this file is essential for the system to load the correct settings and operate as intended.

IRQ - Invalid Regex Query

Criticality	Critical
Location	01_make_accounts.sh
Status	Unresolved

Description

The script is designed to create a new Ethereum account and extract its address from the geth account new output using the following regex:

```shell
ADDR="$(printf '%s\n' "$ADDR_RAW" | grep -oE 'Address:[[:space:]]*\{0x[0-9a-fA-F]+\}' | grep -oE '0x[0-9a-fA-F]+')"
```

However, modern versions of Geth print the address in a different format.As a result, the regex fails to match the account address, leaving the `ADDR` variable empty. This prevents the script from writing the `.coinbase` file, which may break processes relying on the coinbase account.

Recommendation

It is recommended to update the regex to reliably match Ethereum addresses regardless of the surrounding text. This ensures the script correctly extracts the newly created account address and successfully writes it to .coinbase.

EBK - Exposed Bootnode Key

Criticality	Medium
Location	02_make_bootnode_key.sh
Status	Unresolved

Description

The script generates a bootnode private key and saves it to `node/bootnode/boot.key`. This file contains sensitive cryptographic information. Currently, there are no access restrictions on the file or its parent directory. As a result, any user with read access to `boot.key` could impersonate the bootnode, potentially joining the network under false pretenses and disrupting network communication. The lack of restricted permissions on the `node/bootnode` directory further increases the risk of unauthorized access to the private key.

```shell
bootnode -genkey node/bootnode/boot.key
```

Recommendation

It is recommended to restrict access to the bootnode private key and its containing directory using proper file permissions. This ensures that only the owner can read the private key and access the bootnode directory, preventing unauthorized usage or compromise of the bootnode identity.

ECD - Exposed Coinbase Data

Criticality	Medium
Location	01_make_accounts.sh
Status	Unresolved

Description

The script creates a new Ethereum account and stores the account's password in a file (`PASSWORD_FILE`) and the keystore in the validator data directory (`DATADIR`). Currently, there are no restrictions on file or directory permissions, meaning that any user with read access to these files could access the password or keystore and compromise the validator account. This could result in unauthorized access to the account and potential loss of funds.

```shell
mkdir -p "$DATADIR"
touch "$PASSWORD_FILE"
echo "$ADDR" > "$DATADIR/.coinbase"

echo "Coinbase: $ADDR"
```

Recommendation

It is recommended to restrict access to these sensitive files and directories using appropriate permissions. This ensures that only the owner can read the password file and access the validator data directory, improving security and preventing unauthorized access.

DPC - Deprecated Peer Configuration

Criticality	Minor / Informative
Location	03_render_static_nodes.sh
Status	Unresolved

Description

The script `03_render_static_nodes.sh` generates a `configs/static-nodes.json` file that lists peer nodes intended for network connectivity. However, in recent versions of Geth, this file is no longer used for peer discovery or establishing connections. As a result, the generated peer list is ignored by the client, which may lead to nodes failing to connect to the bootnode or to each other unless the required connection parameters are explicitly configured.

```shell
for d in node/validator1 node/validator2 node/rpc;
do
  mkdir -p "$d/geth"
  cp configs/static-nodes.json
"$d/geth/static-nodes.json"

done
```

Recommendation

It is recommended to update the peer configuration to align with current Geth standards. The team should:

- Define peers explicitly in each node's TOML configuration file under the `[P2P]` section using the `StaticNodes` key,
  or
- Pass the peer list during node initialization.

This ensures consistent and reliable peer discovery, improves network connectivity, and enhances overall node synchronization.

EBO - Existent Bootnode Overwrite

Criticality	Minor / Informative
Location	02_make_bootnode_key.sh
Status	Unresolved

Description

The script generates a new bootnode key each time it is executed and writes the resulting address to the `boot.key` file without verifying whether the file already exists. Consequently, repeated executions overwrite the existing `boot.key` file, potentially invalidating references to the original information. This behavior can disrupt account tracking and lead to misconfigured nodes or loss of funds.

```shell
bootnode -genkey node/bootnode/boot.key
ENODE=$(bootnode -nodekey node/bootnode/boot.key
-writeaddress)

echo -n "$ENODE" > node/bootnode/enode.txt
```

Recommendation

It is recommended to check for the existence of `boot.key` before creating a new account and either use the existing coinbase if present or provide an explicit `--force` option to allow overwriting. This ensures that validator accounts remain consistent and prevents accidental overwrites of critical configuration files.

ECO - Existent Coinbase Overwrite

Criticality	Minor / Informative
Location	01_make_accounts.sh
Status	Unresolved

Description

The script generates a new Ethereum account each time it is executed and writes the resulting address to the `.coinbase` file without verifying whether the file already exists. Consequently, repeated executions overwrite the existing `.coinbase` file, potentially invalidating references to the original account. This behavior can disrupt account tracking and lead to misconfigured validator nodes or loss of funds.

```shell
echo "$ADDR" > "$DATADIR/.coinbase"
echo "Coinbase: $ADDR"
```

Recommendation

It is recommended to check for the existence of `.coinbase` before creating a new account and either use the existing coinbase if present or provide an explicit `--force` option to allow overwriting. This ensures that validator accounts remain consistent and prevents accidental overwrites of critical configuration files.

HV - Hardcoded Values

Criticality	Minor / Informative
Location	split.js
Status	Unresolved

Description

The project contains multiple instances where numeric values are directly hardcoded into the code logic rather than being assigned to constant variables with descriptive names. Hardcoding such values can lead to several issues, including reduced code readability, increased risk of errors during updates or maintenance, and difficulty in consistently managing values throughout the codebase.

```shell
var VALIDATORS = [
  "0x014B5831d8a4449Bd8bEbef4540B865a4563Cbd0",
  "0x57d5937C6942B5f59c44c7400834A4946C397734",
  "0x7e96EdCA7866403739E18Fd91582f9A86D4b085a",
  "0xfE4800cA5a48670cc5f591a51F517BA8F682D4Ae"
];
```

Recommendation

It is recommended to replace hardcoded numeric values with variables that represent the actual state of the system. This practice improves code readability and maintainability by clearly indicating the purpose of each value, reducing the likelihood of errors during future modifications.

NPO - Non-Persistent Operations

Criticality	Minor / Informative
Location	split.sh
Status	Unresolved

Description

The `split` script stores essential state variables—such as `_feePool` and `_lastProcessedBlk` in the Geth console's memory. These variables are used to track accumulated fees and the last block processed for distributing rewards to validators, as well as burn, development, and ecosystem addresses. However, because they reside only in memory, restarting the Geth node or console causes these variables to reset. As a result, any fees collected before the restart that have not yet been distributed will be unaccounted for by the script.

Recommendation

To ensure accurate and reliable accounting, it is advised to implement persistent state storage for critical variables such as `_feePool` and `_lastProcessedBlk`. These values should be saved and reloaded on startup to allow the script to resume operation without data loss or inconsistency.

PNI - Public Node Interface

Criticality	Minor / Informative
Location	06_run_validator1.sh 07_run_validator2.sh
Status	Unresolved

Description

The validator script exposes Geth's HTTP and WebSocket RPC interfaces on 0.0.0.0 by default. This binds the APIs to all network interfaces, including public-facing ones if the host has a public IP. Combined with enabled endpoints (eth,net,web3), this can allow external parties to query node state, inspect account balances, observe blockchain activity, or perform denial-of-service attacks. The current configuration is inconsistent with best practices for secure private blockchains.

Shell

```
geth \
  --datadir "$DATADIR" \
  --networkid "$NETWORK_ID" \
  --port "$PORT" \
  --syncmode "full" \
  --gcmode "full" \
  --http --http.addr "$HTTP_ADDR" --http.port "$HTTP_PORT"
\
  --http.api "eth,net,web3" \
  --ws --ws.addr "$HTTP_ADDR" --ws.port "$WS_PORT" --ws.api
"eth,net,web3" \
  --authrpc.addr "$AUTHRPC_ADDR" --authrpc.port
"$AUTHRPC_PORT" --authrpc.jwtsecret "$JWT_FILE" \
  --nat "$NAT" \
  $( [[ -n "$BOOTNODE_ENODE" ]] && echo --bootnodes
"$BOOTNODE_ENODE" ) \
  --mine --miner.etherbase "$COINBASE" \
  --allow-insecure-unlock=false \
  --ipcdisable=false \
```

```
--metrics --pprof.addr "127.0.0.1" --pprof
```

Recommendation

It is recommended to limit access to the node’s RPC interfaces and ensure that only authorized or local clients can connect. Sensitive APIs should remain disabled, and any network exposure should be carefully controlled to reduce the risk of unauthorized access, information leakage, or disruption of the validator node.

USV - Unused State Variables

Criticality	Minor / Informative
Location	06_run_validator1.sh 07_run_validator2.sh
Status	Unresolved

Description

There are state variables that are declared but not used.

```shell
PASSWORD_FILE="${PASSWORD_FILE:-./password.txt}"
...

touch "$PASSWORD_FILE"
```

Recommendation

The team is advised to revise the source code to address redundancies.

ZGT - Zero Genesis Timestamp

Criticality	Minor / Informative
Location	genesis.json
Status	Unresolved

Description

The genesis block in the current configuration has its `timestamp` field set to zero:

```shell
"timestamp": "0x0"
```

While this is technically valid for a private Clique Proof-of-Authority (PoA) network, some Ethereum clients and tooling may interpret a zero timestamp in unexpected ways. This can lead to confusion during network debugging or when using monitoring tools that depend on accurate block timestamps.

Recommendation

It is recommended to set the genesis timestamp to the current Unix time when generating the genesis file. This provides a representative starting point for the network, improves readability for developers and operators, and ensures compatibility with tools that expect non-zero timestamps.

Summary

MCN implements an EVM network based on geth, with multiple validators and a bootnode. Scripts automate account creation, genesis initialization, and network configuration. A JavaScript component collects transaction fees and distributes them to validators, developers, and ecosystem addresses. This audit investigates security issues, business logic, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a TAC blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

A **TAC Security** Company

The Cyberscope team

cyberscope.io