hexens × Camino

SECURITY REVIEW REPORT FOR CAMINO NETWORK

CONFIDENTIAL

CONFIDENTIAL

CONTENTS

ABOUT HEXENS

Hexens is a cybersecurity company that strives to elevate the standards of security in Web 3.0, create a safer environment for users, and ensure mass Web 3.0 adoption.

Hexens has multiple top-notch auditing teams specialized in different fields of information security, showing extreme performance in the most challenging and technically complex tasks, including but not limited to: Infrastructure Audits, Zero Knowledge Proofs / Novel Cryptography, DeFi and NFTs. Hexens not only uses widely known methodologies and flows, but focuses on discovering and introducing new ones on a day-to-day basis.

In 2022, our team announced the closure of a $4.2 million seed round led by IOSG Ventures, the leading web3 venture capital. Other investors include Delta Blockchain Fund, Chapter One, Hash Capital, ImToken Ventures, Tenzor Capital, and angels from Polygon and other blockchain projects.

Since Hexens was founded in 2021, it has had an impressive track record and recognition in the industry: Mudit Gupta - CISO of Polygon Technology - the biggest EVM Ecosystem, joined the company advisory board after completing just a single cooperation iteration. Polygon Technology, Lido, Nubank, TON, 1inch, Coinstats, Hats Finance, Quickswap, Layerswap, 4K, RociFi, as well as dozens of DeFi protocols and bridges, have already become our customers and taken proactive measures towards protecting their assets.

CONFIDENTIAL

LIMITATIONS ON DISCLOSURE AND USAGE OF THIS REPORT

This report has been developed by the company Hexens (the Service Provider) based on the Security review of Camino network (the Client). The document contains vulnerability information and remediation advice.

The information, presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Camino network.

If you are not the intended recipient of this document, remember that any disclosure, copying or dissemination of it is forbidden.

CONFIDENTIAL

AUDIT LED BY

VAHE KARAPETYAN

Co-founder / CTO | Hexens

Audit Starting Date
16.01.2023

Audit Completion Date
21.02.2023

---

hexens × Camino

METHODOLOGY

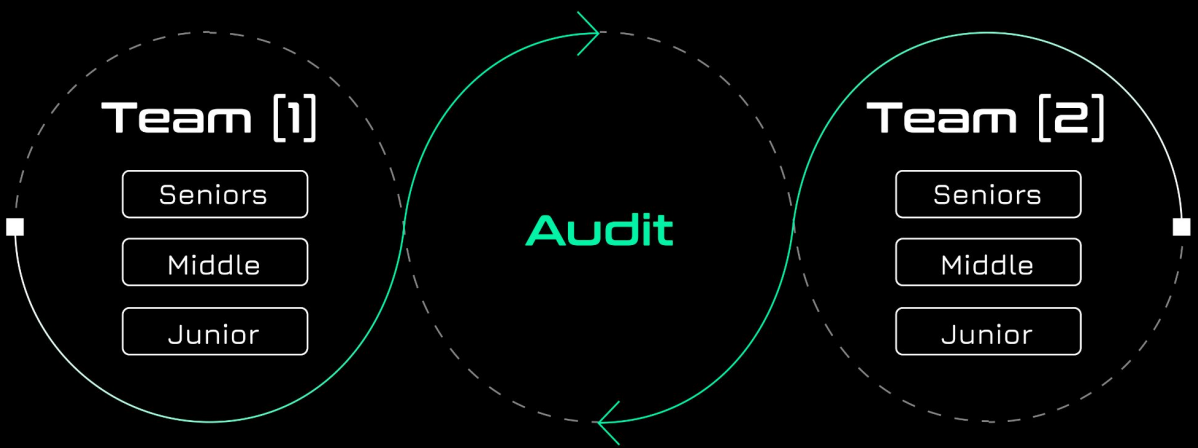
COMMON AUDIT PROCESS

Companies often assign just one engineer to one security assessment with no specified level. Despite the possible impeccable skills of the assigned engineer, it carries risks of the human factor that can affect the product's lifecycle.

Auditor* Audit

HEXENS METHODOLOGY

Hexens methodology involves 2 teams, including multiple auditors of different seniority, with at least 5 security engineers. This unique cross-checking mechanism helps us provide the best quality in the market.



SEVERITY STRUCTURE

The vulnerability severity is calculated based on two components

- Impact of the vulnerability
- Probability of the vulnerability

IMPACT	PROBABILITY			
	Rare	Unlikely	Likely	Very Likely
Low / Info	Low / Info	Low / Info	Medium	Medium
Medium	Low / Info	Medium	Medium	High
High	Medium	Medium	High	Critical
Critical	Medium	High	Critical	Critical

SEVERITY CHARACTERISTICS

Smart contract vulnerabilities can range in severity and impact, and it's important to understand their level of severity in order to prioritize their resolution. Here are the different types of severity levels of smart contract vulnerabilities:

CRITICAL

Vulnerabilities with this level of severity can result in significant financial losses or reputational damage. They often allow an attacker to gain complete control of a contract, directly steal or freeze funds from the contract or users, or permanently block the functionality of a protocol. Examples include infinite mints and governance manipulation.

HIGH

Vulnerabilities with this level of severity can result in some financial losses or reputational damage. They often allow an attacker to directly steal yield from the contract or users, or temporarily freeze funds. Examples include inadequate access control integer overflow/underflow, or logic bugs.

MEDIUM

Vulnerabilities with this level of severity can result in some damage to the protocol or users, without profit for the attacker. They often allow an attacker to exploit a contract to cause harm, but the impact may be limited, such as temporarily blocking the functionality of the protocol. Examples include uninitialized storage pointers and failure to check external calls.

LOW

Vulnerabilities with this level of severity may not result in financial losses or significant harm. They may, however, impact the usability or reliability of a contract. Examples include slippage and front-running, or minor logic bugs.

INFORMATIONAL

Vulnerabilities with this level of severity are regarding gas optimizations and code style. They often involve issues with documentation, incorrect usage of EIP standards, best practices for saving gas, or the overall design of a contract. Examples include not conforming to ERC20, or disagreement between documentation and code.

It's important to consider all types of vulnerabilities, including informational ones, when assessing the security of smart contracts. A comprehensive security audit should consider all types of vulnerabilities to ensure the highest level of security and reliability.

SCOPE

The analyzed resources are located on:
https://github.com/chain4travel/caminogo
https://github.com/chain4travel/camino-node
https://github.com/chain4travel/camino-wallet

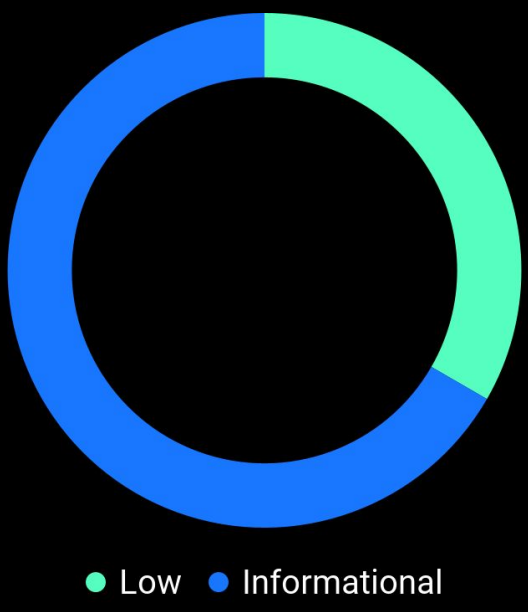
CONFIDENTIAL

SUMMARY

SEVERITY	NUMBER OF FINDINGS
CRITICAL	0
HIGH	0
MEDIUM	0
LOW	3
INFORMATIONAL	6

TOTAL: 9

SEVERITY



STATUS

WEAKNESSES

This section contains the list of discovered weaknesses.

1. AN ADMIN CAN REMOVE ADDRESSSTATEROLEADMINBIT FROM THEMSELVES AND OTHER ADMINISTRATORS

SEVERITY: Low

PATH: [caminogo]. caminogo/vms/platformvm/txs/executor/camino_tx_executor.go

REMEDIATION: disallow to remove the admin bit from the sender's account or if there are no other admins

STATUS:

DESCRIPTION:

The **AddressStateTx** handler allows a user to remove the admin bit from themselves. If the only administrative account does this, there will be no remaining admins.

2. AN ADMIN CAN REVOKE ROLES FROM THEMSELVES IN THE C-CHAIN

SEVERITY: Low

PATH: [caminoethvm]. caminoethvm/contracts/access.sol

REMEDIATION: disallow to remove the admin bit from the sender's account or if there are no other admins

STATUS:

DESCRIPTION:

There is a revokeRole() function that can be called by an admin to remove roles from the account.

There is no check that the admin is not removing the admin role from himself. If the only administrative account does this, there will be no remaining admins.

3. POTENTIAL NODE API DOS

SEVERITY: Low

PATH: [caminogo]. vms/platformvm/service.go, vms/platformvm/camino_service.go

REMEDIATION: implement the default paging of the API

STATUS:

DESCRIPTION:

Number of API endpoints of the node might be susceptible to DoS attacks. The denial of service may be caused by either a huge response size amplification or by heavy computation on the input data.

Examples of such API calls are **platform.getCurrentValidators** (proved to cause significant load and delays on the AVAX nodes) and **platform.getClaimables** (accepts an unbounded array of deposit tx ids).

4. IMPORTTX DOES NOT VERIFY IF IMPORTEDINPUTS ARE LOCKED

SEVERITY: Informational

PATH: [caminogo]. caminogo/vms/platformvm/txs/executor/camino_tx_executor.go

REMEDIATION: add the following line in the ImportTx function in camino_tx_executor.go before executing transaction:

```
if err := locked.VerifyNoLocks(tx.ImportedInputs); err != nil {     return err
}
```

STATUS:

DESCRIPTION:

An executor of the **ImportTx** transaction does not verify that tx.ImportedInputs does not have locked inputs via **locked.VerifyNoLocks**. Although it does not seem possible to export a locked UTXO from C-Chain at the moment of audit due to lack of **locked.In** and **locked.Out** definitions in C-Chain transaction codec, it might be possible to do so in future.

5. ANYBODY CAN SIGN REWARDSIMPORTTX

SEVERITY: Informational

PATH: [caminogo]. caminogo/vms/platformvm/txs/executor/camino_tx_executor.go

REMEDIATION: review the rewards import procedure in order to make it possible to authorize the transactions coming from the mempool

STATUS:

DESCRIPTION:

The executor of **RewardsImportTx** does not verify credentials of transaction inputs which makes it possible for anyone to submit this transaction to the mempool. Although UTXO **outputOwners** can only be the treasury address, it might be worth investigating possible ways to authorize the submission of this transaction.

6. ANYBODY CAN SIGN UNLOCKDEPOSITTX FOR EXPIRED DEPOSITS

SEVERITY: Informational

PATH: [caminogo]. caminogo/vms/platformvm/utxo/camino_locked.go

REMEDIATION: review the deposit unlock procedure in order to make it possible to authorize the transactions coming from the mempool

STATUS:

DESCRIPTION:

The function **VerifyUnlockDepositedUTXOs** verifies transaction credentials against inputs via **VerifyMultisigTransfer** only for non-deposit UTXOs.

```
if isDeposited = lockedOut.DepositTxID != ids.Empty; !isDeposited {
  return nil, errUnlockingUnlockedUTXO
}
/* ... */
if isDeposited {
  /* this branch lacks VerifyMultisigTransfer */
} else {
        if err := h.fx.VerifyMultisigTransfer(tx, in, creds[index], out, h.utxosReader); err != nil {
           return nil, fmt.Errorf("failed to verify transfer: %w", err)
/* ...*/
        }
}
```

Thus, it is possible to unlock an expired deposit for any account. Test case for camino_locked.go:

```
"UTXO outputOwners do not correspond to creds2": {
  args: args{
    chainState: func(ctrl *gomock.Controller) state.Chain {
      s := state.NewMockChain(ctrl)
      s.EXPECT().GetTimestamp().Return(depositExpiredTime)
      s.EXPECT().GetDeposit(depositID).Return(deposit1, nil)
      s.EXPECT().GetDepositOffer(deposit1.DepositOfferID).Return(depositOffer, nil)
      return s
    },
    tx: tx,
    utxos: []*avax.UTXO{
      generateTestUTXO(ids.ID{9, 9}, assetID, deposit1.Amount, outputOwners, depositID,
ids.Empty),
    },
    ins: []*avax.TransferableInput{
      generateTestIn(assetID, deposit1.Amount, depositID, ids.Empty, sigIndices),
    },
    outs: []*avax.TransferableOutput{
      generateTestOut(assetID, deposit1.Amount, outputOwners, ids.Empty, ids.Empty),
    },
    creds:       []verify.Verifiable{cred2},
    burnedAmount: 0,
    assetID:     assetID,
  },
  want: map[ids.ID]uint64{depositID: deposit1.Amount},
},
```

7. TABNABBING IS POSSIBLE FOR THE OLD BROWSERS

SEVERITY: Informational

PATH: [camino-wallet]

REMEDIATION: don't render links with the javascript: schema and explicitly specify the rel="noopener noreferrer" attribute

STATUS:

DESCRIPTION:

Users can control the href attribute of the **<a>** tag in NFTs that contains links. This tag is supporting **javascript**: protocol scheme and can be used by attackers to perform tabnabbing or XSS attacks.

However the **<a>** tags have the **target="_blank"** attribute which provides the same **rel** behavior as setting **rel="noopener"**. But if a user uses an old version of the browser users can be tricked by the attacker.

8. IMPOSSIBLE TO ADD ERC1155 TOKENS TO THE WALLET

SEVERITY: Informational

PATH: [camino-wallet]

REMEDIATION: correctly implement the interface to the ERC1155 tokens in the wallet

STATUS:

DESCRIPTION:

During the adding of the NFT tokens wallet is trying to read the symbol and name of the submitted contract via calling **symbol()** and **name()** functions. ERC1155 tokens don't have these getters and these calls are reverted.

9. NFTS REMAIN IN THE WALLET AFTER TRANSFER

SEVERITY: Informational

PATH: [camino-wallet]

REMEDIATION: correctly implement the NFT ownership detection

STATUS:

DESCRIPTION:

Avalanche wallet uses **ERC721Enumerable** to track current owners of an NFTs. Camino wallet collects owners of NFTs by parsing events. However, the wallet shows all tokens that were transferred to a user, even those that were already sent to another address and don't belong to the current user.