SECURITY REVIEW OF VOLMEX

hyacinth

SECURITY REVIEW VOLMEX

Summary

Auditors: 0xWeiss

Marketplace: Hyancinth

Client: Volmex

Report Delivered: August 2024

Protocol Summary

Protocol Name	Volmex
Language	Solidity
Codebase	https://github.com/volmexfinance/token
Commit	6894013e601be458c57f5bde6f88290f99588225
Previous Audits	No

About 0xWeiss

Marc Weiss, or **0xWeiss**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Reach out on Twitter @0xWeisss or on Telegram @0xWeiss.

Audit Summary

Volmex engaged 0xWeiss through Hyacinth Audits to review the security of its token contract. From the 12th of August to the 13th of August, 0xWeiss reviewed the source code in scope. At the end, there were 4 issues identified. All findings have been recorded in the following report. Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

Vulnerability Summary

Severity	Total	Pending	Acknowledged	Par. resolved	Resolved
HIGH	0	0	0	0	0
MEDIUM	0	0	0	0	0
LOW	1	0	0	0	1
INF	3	0	0	0	3

Audit Scope

CORE

ID	File Path
VOL	contracts/Volmex.sol

Severity Classification

Severity	Classification
HIGH	Exploitable, causing loss/manipulation of assets or data.
MEDIUM	Risk of future exploits that may or may not impact the smart contract execution.
LOW	Minor code errors that may or may not impact the smart contract execution.
INF	No impact issues. Code improvement

Methodology

The auditing process pays special attention to the following considerations:

● Testing the smart contracts against both common and uncommon attack vectors.

● Assessing the codebase to ensure compliance with current best practices and industry standards.

● Ensuring contract logic meets the specifications and intentions of the client.

● Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.

● Thorough line-by-line manual review of the entire codebase by industry experts.

Findings and Resolutions

ID	Category	Severity	Status
VOL-1	Missing checks	**LOW**	Resolved
VOL-2	Incorrect Naming	**INF**	Resolved
VOL-3	Redundancy	**INF**	Resolved
VOL-4	Centralization	**INF**	Resolved

VOL-1 | No max supply in case of private key leakage

Severity	Category	Status
LOW	Missing checks	Resolved

Description of the issue

Currently, the minting of the tokens is structured so that any address holding the `MINTER_ROLE` would be able to mint as many tokens as they would want.

In case that any of the `MINTER_ROLE` private key gets leaked, they would be able to mint as many tokens as they would want and dilute the value of the token:

```
function setMinterRole(address minter) external {
    _requireAdminRole();
    _grantRole(MINTER_ROLE, minter);
  }

  function mint(address to, uint256 amount) external {
    _requireMinterRole();
    _mint(to, amount);
  }
```

Recommendation

Always use multi-signature wallets for roles, and add a max supply variable and enforce it inside the `mint` function.

Resolution

Fixed at PR

VOL-2 | Incorrect parameter naming when burning tokens

Severity	Category	Status
INF	Incorrect Naming	Resolved

Description of the issue

When calling the `burn` function, the address where you burn the tokens from is called `to`, while in reality it is the address you are burning the tokens `from`:

```
function burn(address to, uint256 amount) external {
    _requireBurnerRole();
    _burn(to, amount);
  }
```

Recommendation

Update the naming:

```
 - function burn(address to, uint256 amount) external {
+ function burn(address from, uint256 amount) external {
    _requireBurnerRole();
    _burn(to, amount);
  }
```

Resolution

Fixed at PR

VOL-3 | Function redundancy to check roles

Severity	Category	Status
INF	Redundancy	Resolved

Description of the issue

The Volmex token contract uses `AccessControlUpgradeable` from OZ, which allows to grant roles and check permissions across the contract. `AccessControlUpgradeable` already has a function that check whether `msg.sender` has a role or not:

```
function _checkRole(bytes32 role, address account) internal view virtual {
    if (!hasRole(role, account)) {
        revert AccessControlUnauthorizedAccount(account, role);
    }
}
```

but, Volmex still declares its own functions, while it is not needed:

```
function _requireAdminRole() internal view {
    require(hasRole(ADMIN_ROLE, msg.sender), "VOL: Not Admin");
}
function _requireMinterRole() internal view {
    require(hasRole(MINTER_ROLE, msg.sender), "VOL: Not Minter role");
}
function _requireBurnerRole() internal view {
    require(hasRole(BURNER_ROLE, msg.sender), "VOL: Not Burner role");
}
```

Recommendation

Use the `_checkRole` function instead

Resolution

Fixed at PR

VOL-4 | Centralization risks

Severity	Category	Status
INF	Centralization	Resolved

Description of the issue

Currently, the Volmex token allows the owners of specific roles to mint and burn tokens from a specific address. While this is the intended functionality, extreme caution has to be taken when dealing with such roles.

```
function mint(address to, uint256 amount) external {
    _requireMinterRole();
    _mint(to, amount);
}
function burn(address to, uint256 amount) external {
    _requireBurnerRole();
    _burn(to, amount);
}
```

Recommendation

Make sure the roles are correctly documented and a correct wallet structure has been adopted to handle key roles of the architecture.

Resolution

These roles will be assigned to multisig wallets.

DISCLAIMER

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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Marc Weiss to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

My position is that each company and individual are responsible for their own due diligence and continuous security. My 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. Therefore, I do not guarantee the explicit security of the audited smart contract, regardless of the verdict.