SALUS SECURITY

DEC 2025

CODE SECURITY ASSESSMENT

KAPKAP

Overview

Project Summary

- Name:Kapkap
- Platform: OpBNB Network
- Language: Solidity
- Address:
    - https://github.com/dfun-inc/contract/
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Kapkap
Version	v2
Type	Solidity
Dates	Dec 14 2025
Logs	Dec 13 2025; Dec 14 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	3
Total informational issues	3
Total	6

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Token holders cannot burn their own ERC1155 unless they are whitelisted as a minter	Low	Business Logic	Resolved
2	Centralization risk	Low	Centralization	Acknowledged
3	TokensOfOwner returns global IDs and may revert on page boundaries	Low	Business Logic	Resolved
4	Missing events for functions that change critical state	Informational	Logging	Resolved
5	Error message is incorrect	Informational	Logging	Resolved
6	File and contract names mismatch	Informational	Configuration	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Token holders cannot burn their own ERC1155 unless they are whitelisted as a minter	
Severity: Low	Category: Business Logic
Target: - src/KapKapCenter.sol	

Description

`NFT1155.burn()` is restricted by `onlyMinter()`, meaning only addresses whitelisted in `_minters` can initiate burns. Even if a user owns the tokens, they cannot burn them unless they are also a `minter`. The additional check `(account == msg.sender || isApprovedForAll(account, msg.sender))` only allows a `minters` to burn its own tokens, or to burn a holder's tokens if the holder has granted operator approval to that `minters`.

src/KapKapCenter.sol:L150-L155

```
function burn(address account, uint256 tokenId, uint256 amount) external override
onlyMinter
    {
    require( account == msg.sender ||  isApprovedForAll(account,msg.sender), "ERC1155:
burn caller is not owner nor approved");
    _burn(account, tokenId, amount);

}
```

Recommendation

Remove the `onlyMinter()` restriction from `burn()/burnBatch()`.

Status

The team has resolved this issue in commit 297507b.

2. Centralization risk	
Severity: Low	Category: Centralization
Target: - src/KapKapCenter.sol	

Description

In the `` `KapKapCenter` `` contract, there exists one privileged role `` `minter` ``. The `` `minter` `` has the authority to some key functions such as `` `mint()` ``, `` `mintBatch()` ``, `` `burn()` `` and `` `burnBatch()` ``. If the role's private key is compromised, an attacker could trigger this function to freely mint or burn NFTs.

src/KapKapCenter.sol:L117-L167

```
function mint(address account, uint256 tokenId, uint256 amount, bytes memory data)
external override onlyMinter
function mintBatch(address account, uint256[] memory tokenIds, uint256[] memory amounts,
bytes memory data) external override onlyMinter
function burn(address account, uint256 tokenId, uint256 amount) external override
onlyMinter
function burnBatch(address account, uint256[] memory tokenIds, uint256[] memory amounts)
external override onlyMinter
```

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been resolved by the team and states that due to the business requirement, the one privileged role can’t be removed now.

3. TokensOfOwner returns global IDs and may revert on page boundaries	
Severity: Low	Category: Business Logic
Target: - src/KapKapCenter.sol	

Description

The `tokensOfOwner()` uses `offset` as a page index (`start = offset * pageMax`) over the global `_mintedIds` set. The bounds logic allows `offset == balance / pageMax`, but when `balance % pageMax == 0`, the function computes `maxCount = 0` and reverts, causing unexpected failures for indexers/frontends iterating pages.

For example, if `_mintedIds.length() = 20` and `pageMax = 10`, then `pages = 20 / 10 = 2`. Calling `tokensOfOwner()` passes `require(pages >= offset)` but then hits `maxCount = 20 % 10 = 0` and reverts.Separately, even for `offset=0`, the function returns 10 minted IDs and their balances; if the owner holds none of them, the returned list will contain many entries with `amount = 0`.

Additionally, the function name is misleading: it does not return only IDs owned by owner; it returns a slice of all minted IDs with `balanceOf(owner, id)`, often including many entries with `amount = 0`.

src/KapKapCenter.sol:L179-L208

```
function tokensOfOwner(address owner, uint256 offset, uint256 pageMax) ... {
   ...
   if (balance <= pageMax) {
       maxCount = balance;
   } else {
       maxCount = pageMax;
       uint256 pages = balance / pageMax;
       require(pages >= offset, "invalid page size!");
       if (pages == offset) {
           maxCount = balance % pageMax;
           require(maxCount > 0, "invalid page size!");}
     }
   nftInfos = new INFT1155.NftInfo1155[](maxCount);
   for (uint256 i = 0; i < maxCount; i++) {
       nftInfos[i].id = _mintedIds.at(offset * pageMax + i);
       nftInfos[i].amount = balanceOf(owner, _mintedIds.at(offset * pageMax + i));
}}
```

Recommendation

Optimize the pagination logic to avoid empty-page reverts.

Status

The team has resolved this issue in commit 297507b.

2.3 Informational Findings

4. Missing events for functions that change critical state	
Severity: Informational	Category: Logging
Target: - src/KapKapCenter.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the `` `NFT1155` `` contract, events are lacking in the privileged functions (e.g. `` `addMinter()` ``, `` `removeMinter()` ``, `` `setMetaType()` ``) and `` `setURI()` ``.

In the `` `KapKapCenter` `` contract, events are lacking in the privileged functions (e.g. `` `addPayMethod()` `` and `` `removePayMethod()` ``.

Recommendation

It is recommended to emit events for critical state changes.

Status

The team has resolved this issue in commit 297507b.

5. Error message is incorrect	
Severity: Informational	Category: Logging
Target: - src/KapKapCenter.sol	

Description

When a caller attempts to query a URI for a non-existent tokenId, the contract throws an exception. The original error message would clearly state `ERC1155Metadata:`, but the current error message misleads developers or users.

src/KapKapCenter.sol:L95-L99

```
function uri(uint256 tokenId) public view virtual override returns (string memory) {
    require(_mintedIds.contains(tokenId), "ERC721Metadata: burn caller is not owner nor
approved");
    return bytes(_baseURI).length > 0 ? string(abi.encodePacked(_baseURI,
tokenId.toString(), _metatype)) : "";
}
```

Recommendation

Use `ERC1155Metadata` instead of `ERC721Metadata`

Status

The team has resolved this issue in commit 297507b.

6. File and contract names mismatch	
Severity: Informational	Category: Configuration
Target: - src/KapKapCenter.sol	

Description

Some files in the codebase have names that do not match the contract defined within. For example, the file `KapKapCenter` contains a contract named `INFT1155`, `NFT1155` ,`INFT721` and `NFT721`. Such mismatches can create confusion for developers and auditors, reducing code readability and maintainability.

Recommendation

It is recommended to rename the file to match the contract name.

Status

The team has resolved this issue in commit 297507b.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 0e191da:

File	SHA-1 hash
NFT1155.sol	d5f6f50b3f6118b82c193f9c1cebd16a02093020
KapKapCenter.sol	026218a89db8daa82ca083ed9e1ed53d5404b44e