Competitive Security Assessment

Project Twelve (P12)

Aug 29th, 2022

Secure3

secure3.io

Table of Contents

Summary

P12 is a GameFi platform with ERC20 game coin creation, ERC1155 game NFT creation, game NFT marketplace, staking liquidity mining features. It connects gamers, game developers, liquidity providers in the game platform and tokenomics.

This report has been prepared for P12 to identify issues and vulnerabilities in the smart contract source code of the ProjectTwelve project. A comprehensive examination with Static Analysis and Manual Review techniques has been performed.

The examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static scanner to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Informational, Medium, Critical. For each of the findings we have provided recommendation of a fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	ProjectTwelve / P12
Platform & Language	Ethereum, Solidity
Codebase	https://github.com/ProjectTwelve/contracts audit commit - 1703571816f04bb51ca689d64692c7035e45324d final commit - 8de3f9fab1eed71afeaf4b8a55ec21f38856a76c
Audit Methodology	• Business Logic Understanding and Review • Privileged Roles Review • Static Analysis • Code Review

Business Logic Review Summary

Total Number of Features	Caution	Information	Verified
10	0	1	9

Privileged Role Review Summary

Total Number of Privileged Roles	Caution	Information	Verified
14	0	2	12

Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowleged	Fixed	Mitigated
Critical	1	0	0	1	0
Medium	6	0	3	3	0
Informational	15	0	4	11	0

Audit Scope

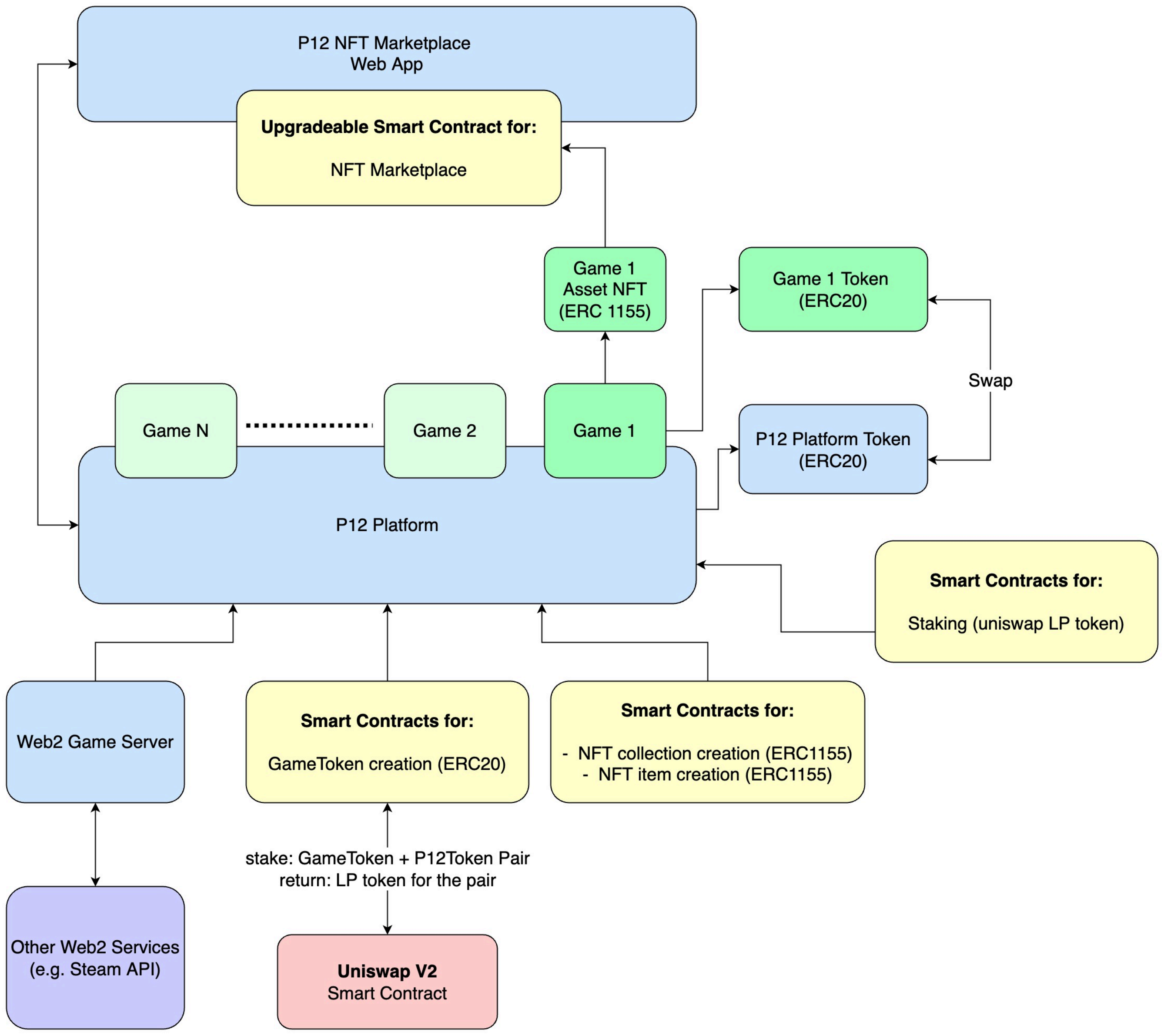
File	Commit Hash
contracts/factory/P12V0ERC20.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/P12V0FactoryStorage.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/P12V0FactoryUpgradeable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/interfaces/IP12Mine.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/interfaces/IP12V0ERC20.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/interfaces/IP12V0Factory.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/factory/interfaces/IP12V0FactoryUpgradeable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/ERC1155Delegate.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/MarketConsts.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/SecretShopStorage.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/SecretShopUpgradable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/interfaces/IDelegate.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/interfaces/ISecretShopUpgradable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/secretShop/interfaces/IWETHUpgradable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/sft-factory/P12Asset.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/sft-factory/P12AssetFactoryStorage.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/sft-factory/P12AssetFactoryUpgradable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/sft-factory/interfaces/IP12Asset.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/sft-factory/interfaces/IP12AssetFactoryUpgradable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/staking/P12MineStorage.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/staking/P12MineUpgradeable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/staking/P12RewardVault.sol	1703571816f04bb51ca689d64692c7035e45324d

File	Commit Hash
contracts/staking/interfaces/IP12MineUpgradeable.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/staking/interfaces/IP12RewardVault.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/token/IP12Token.sol	1703571816f04bb51ca689d64692c7035e45324d
contracts/token/P12Token.sol	1703571816f04bb51ca689d64692c7035e45324d

Business Logic Review

In this section, we asked project team to provide a list of business features of their contracts, our team verified each feature one by one and provided the verification results below.

This diagram illustrates the P12 on-chain and off-chain eco system.



How to read the table

1. **Left column is from project team**, describing their business intend
2. **Right column is from auditing team**, verifying if the code implementation meets the claimed business intend

Business Feature Claimed	Business Feature Audit Result
Game ERC20 Token - Game developer can register and publish game coin. The game coin should be bind to a game.	◉ **Auditor Evaluation: Verified** ◉ **Code Reference:** ◉ contracts/contracts/factory/P12V0ERC20.sol:27 ◉ Contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:127,142,198 ◉ **Detail:** The game token contract inherits **ERC20 and ERC20Burnable** standard. The `register()` and `create()` function in `P12V0FactoryUpgradeable` takes `gameId` and `developer` address to create a game coin (ERC20) and register to the developer.
Game ERC20 Token - Game developer can configure coin with game coin full name, symbol, game coin icon, total supply, distribution between P12 and the game coin	◉ **Auditor Evaluation: Information** ◉ **Code Reference:** ◉ contracts/contracts/factory/P12V0ERC20.sol:27 ◉ Contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:142,153,198 ◉ **Detail:** The game token contract `P12V0ERC20` takes `name_`, `symbol_`, `gameCoinIconUrl_`, `amount_` as the input parameter for developer to configure the attributes. `P12V0FactoryUpgradeable::create()` takes `amountGameCoin` and `amountP12` but **only half (L153) goes to the Uniswap liquidity pool**. Reader should be aware of the distribution.
Gamer can withdraw game coin with amount to EOA	◉ **Auditor Evaluation: Verified** ◉ **Code Reference:** Contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:273 ◉ **Detail:** The `P12V0FactoryUpgradeable::withdraw()` function transfers game coin to the address provided. The function is only callable by the contract owner `onlyOwner`.

Business Feature Claimed	Business Feature Audit Result
ERC1155 Token - Allow create and mint a new asset.	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/sft-factory/P12Asset.sol:46,59 ◉ **Detail:** `create()` and `mint()` function in `P12Asset` create and mint specific amount of tokens to an address.
Game Developer can create a new **NFT collection** with logo image, featured image, collection name and description	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/sft-factory/P12AssetFactoryUpgradable.sol:59 ◉ **Detail:** The `P12AssetFactoryUpgradable::createCollection()` function takes `contractURI` and create a `P12Asset` as collection. Other attributes are stored off chain.
Game Developer can create a new **NFT item** with image, name, NFT collection name, GUID, amount of the NFT, attributes, description.	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/sft-factory/P12AssetFactoryUpgradable.sol:78 ◉ **Detail:** The `P12AssetFactoryUpgradable::createAssetAndMint()` function takes `collection`, `amount`, `uri` parameter to create and mint a NFT token. Other attributes are stored off chain.
User can buy a NFT token listed by the seller	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** ◉ contracts/contracts/secretShop/ERC1155Delegate.sol:85 ◉ Contracts/contracts/secretShop/SecretShopUpgradable.sol:226 ◉ **Detail:** The `SecretShopUpgradable::run()` function verifies the order signature and transfer the token from seller to buyer address.
Seller can delist a listed NFT token	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/secretShop/SecretShopUpgradable.sol:347 ◉ **Detail:** The function updates the `inventoryStatus` to `InvStatus.CANCELLED` when user's operation is cancel offer

Business Feature Claimed	Business Feature Audit Result
User can stake LP token and earn reward	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/staking/P12MineUpgradeable.sol:286,340,356 ◉ **Detail:** The `P12MineUpgradeable::deposit()` function takes `lpToken` address and `amount` to transfer from message sender to `P12MineUpgradeable` contract. The `claim()` and `claimAll()` function allow user to get reward for a specific liquidity pool or all the pools that the address has stake.
User can withdraw LP token staked	◉ **Auditor Evaluation:** **Verified** ◉ **Code Reference:** Contracts/contracts/staking/P12MineUpgradeable.sol:378 ◉ **Detail:** The `P12MineUpgradeable::withdraw()` function takes `pledger` address `lpToken` address and withdraw `id` to transfer LP token staked to `pledger` address.

Privilege Role Review

In this section, we reviewed all the privileged roles in the contracts. We listed all the findings in the following table.

How to read the table

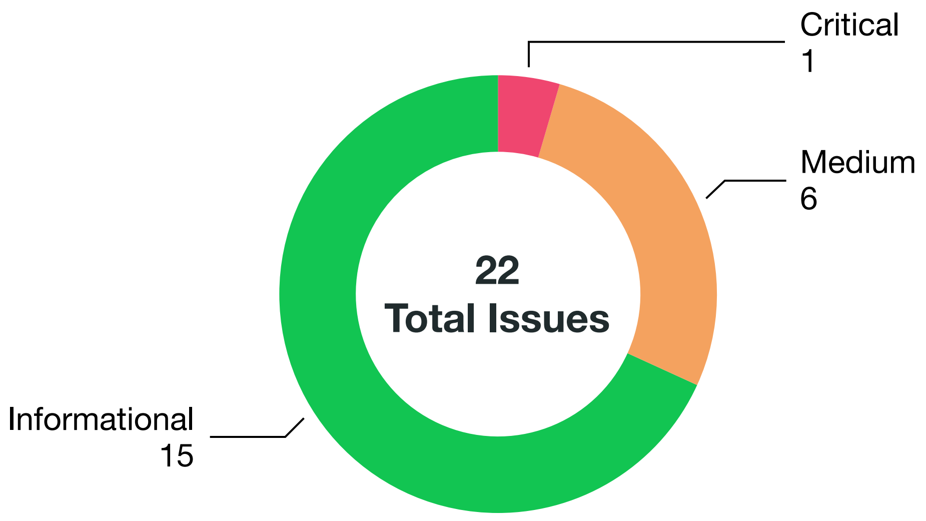
1. **Left column:** privileged role name
2. **Middle column:** privileged permission of the role
3. **Right column:** verified code implementation and roles permission by auditing team

Contract Role	Privileged Functionalities	Audit Review
P12V0ERC20 Owner Address	◉ mint	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/factory/**P12V0ERC20.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12V0FactoryUpgradeable Owner Address	◉ withdraw ◉ setDelayK ◉ setDelayB ◉ register ◉ setP12Mine ◉ pause ◉ unpause	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/factory/**P12V0FactoryUpgradeable.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12V0FactoryUpgradeable Game Developer Address	◉ create ◉ declareMintCoin	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/factory/**P12V0FactoryUpgradeable.sol** ◉ **Detail:** critical functionalities can only be called by game developer

Contract Role	Privileged Functionalities	Audit Review
ERC1155Delegate DELEGATION_CALLER Role Address	◉ executeSell	◉ **Auditor Evaluation: Information,** ◉ **Code Reference:** contracts/contracts/secretShop/**ERC1155Delegate.sol** ◉ **Detail:** critical functionalities can only be called by **DELEGATION_CALLER** role. However, contract owner needs to manually grant address for this role by manually calling `grantRole()` function after deployment. The address will be chosen by the contractor owner.
ERC1155Delegate PAUSABLE_CALLER Role Address	◉ pause ◉ unpause	◉ **Auditor Evaluation: Information,** ◉ **Code Reference:** contracts/contracts/secretShop/**ERC1155Delegate.sol** ◉ **Detail:** critical functionalities can only be called by **PAUSABLE_CALLER** role. However, contract owner needs to manually grant address for this role by manually calling `grantRole()` function after deployment. The address will be chosen by the contractor owner.
SecretShopUpgradable Owner Address	◉ pause ◉ unpause ◉ updateFeeCap ◉ updateDelegates ◉ updateCurrencies	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/secretShop/**SecretShopUpgradable.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12Asset Owner Address	◉ create ◉ mint ◉ setContractURI	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/sft-factory/**P12Asset.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12AssetFactoryUpgradable Owner Address	◉ pause ◉ unpause	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/sft-factory/**P12AssetFactoryUpgradable.sol** ◉ **Detail:** critical functionalities can only be called by contract owner

Contract Role	Privileged Functionalities	Audit Review
P12AssetFactoryUpgradable NFT **Collection Developer** Owner Address	◉ createAssetAndMint ◉ updateCollectionUri	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/sft-factory/**P12AssetFactoryUpgradable.sol** ◉ **Detail:** critical functionalities can only be called by NFT collection developer owner
P12AssetFactoryUpgradable Game Developer Address	◉ createCollection	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/sft-factory/**P12AssetFactoryUpgradable.sol** ◉ **Detail:** critical functionalities can only be called by game developer
P12MineUpgradeable Owner Address	◉ pause ◉ unpause ◉ setReward ◉ setDelayK ◉ setDelayB ◉ createPool	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/staking/**P12MineUpgradeable.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12Factory Contract Address	◉ createPool ◉ addLpTokenInfoForGameCreator	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/staking/**P12MineUpgradeable.sol** ◉ **Detail:** critical functionalities can only be called by **P12Factory** contract address
P12RewardVault Owner Address	◉ reward	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/staking/**P12RewardVault.sol** ◉ **Detail:** critical functionalities can only be called by contract owner
P12Token Owner Address	◉ mint	◉ **Auditor Evaluation: Verified,** ◉ **Code Reference:** contracts/contracts/token/**P12Token.sol** ◉ **Detail:** critical functionalities can only be called by contract owner

Code Assessment Findings



ID	Name	Category	Severity	Status
P12-1	Call initialize() function as soon as possible	Privilege Related	Informational	Acknowledged
P12-2	P12V0ERC20::transferWithAccount() unchecked transfer return value	Logical Issue	Medium	Acknowledged
P12-3	P12V0FactoryUpgradeable::initialize() does not validate input address	Logical Issue	Informational	Fixed
P12-4	After deployment owner cannot update P12V0FactoryUpgradeable.addLiquidityEffectiveTime	Logical Issue	Medium	Acknowledged
P12-5	P12V0FactoryUpgradeable P12 token approved uniswapRouter with uint256 max amount	Privilege Related	Informational	Acknowledged

ID	Name	Category	Severity	Status
P12-6	P12V0FactoryUpgradeable mixed use of SafeMath and arithmetic operators hurts readability	Code Style	Informational	Fixed
P12-7	P12V0FactoryUpgradeable::register() developer address is not checked	Logical Issue	Informational	Fixed
P12-8	P12V0FactoryUpgradeable p12mine can be used before assign a value	Logical Issue	Medium	Fixed
P12-9	P12V0FactoryUpgradeable::executeMint() issue with non-existent gameCoinAddress and mintId	Logical Issue	Critical	Fixed
P12-10	P12V0FactoryUpgradeable::declareMintCoin() logical issue with mintId	Logical Issue	Informational	Acknowledged
P12-11	Pair.salt is not used	Code Style	Informational	Acknowledged
P12-12	Op.COMPLETE_BUY_OFFER is not used	Code Style	Informational	Fixed
P12-13	SettleShared.amountToEth is not used	Code Style	Informational	Fixed
P12-14	Unused local variable amountETH in SecretShopUpgradable	Code Style	Informational	Fixed
P12-15	P12Asset::mint() logical issue when id does not exist	Logical Issue	Medium	Fixed
P12-16	P12Asset::_setUri() logical issue with empty string	Logical Issue	Informational	Fixed
P12-17	P12Asset::mint() reentrancy risk to bypass maxSupply	Logical Issue	Medium	Fixed
P12-18	P12AssetFactoryUpgradable::initialize() does not validate address	Logical Issue	Informational	Fixed
P12-19	No contractURI validation in P12AssetFactoryUpgradable::createCollection()	Logical Issue	Informational	Fixed
P12-20	P12MineUpgradeable::initialize() does not validate address	Logical Issue	Informational	Fixed
P12-21	P12RewardVault::constructor() does not validate address	Logical Issue	Informational	Fixed
P12-22	P12Token unlimited mint	Privilege Related	Medium	Acknowledged

P12-1: Call `initialize()` function as soon as possible

Category	Severity	Code Reference	Status
Privilege Related	Informational	• contracts/contracts/factory/P12V0FactoryUpgradeable.sol:42 • contracts/contracts/secretShop/SecretShopUpgradable.sol:54 • contracts/contracts/sft-factory/P12AssetFactoryUpgradable.sol:36 • contracts/contracts/staking/P12MineUpgradeable.sol:50	Acknowledged

Code

```
function initialize(…) public initializer {}
```

Description

With proxy and upgradeable pattern, the contract owner needs to call `initialize()` function manually. Otherwise, the contract will remain as uninitialized state and anyone can call it and take the ownership of the contract.

Recommendation

Manually call the `initialize()` as soon as possible after deployment.

Client Response

Acknowledged. The contracts will be deployed with OZ upgrade plugin, which will atomically call `initialize()` function after deploying `Proxy` contract.

P12-2: `P12V0ERC20::transferWithAccount()` unchecked transfer return value

Category	Severity	Code Reference	Status
Logical Issue	Medium	contracts/contracts/factory/P12V0ERC20.sol:60	Acknowledged

Code

```
60:     transfer(recipient, amount);
61:     emit TransferWithAccount(recipient, account, amount);
```

Description

The ERC20 `transfer()` function has a return value, and in case of failure it returns false. The best practice is to check the return value of the `transfer()` function and revert in case of failure.

Recommendation

Check the return value

Client Response

Acknowledged. This is safe because `transfer()` is only calling the contract itself.

P12-3: `P12V0FactoryUpgradeable::initialize()` does not validate input address

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:49-51	Fixed

Code

```
42:   function initialize(
43:     address p12_,
44:     address uniswapFactory_,
45:     address uniswapRouter_,
46:     uint256 effectiveTime_,
47:     bytes32 initHash_
48:   ) public initializer {
49:     p12 = p12_;
50:     uniswapFactory = uniswapFactory_;
51:     uniswapRouter = uniswapRouter_;
52:     _initHash = initHash_;
53:     addLiquidityEffectiveTime = effectiveTime_;
```

Description

The input parameter `p12_`, `uniswapFactory_` and `uniswapRouter_` three input address could be zero.

Recommendation

Validate `p12_`, `uniswapFactory_` and `uniswapRouter_` is not `address(0)`.

Client Response

Fixed.

P12-4: After deployment owner cannot update `P12V0FactoryUpgradeable.addLiquidityEffectiveTime`

Category	Severity	Code Reference	Status
Logical Issue	Medium	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:168	Acknowledged

Code

```
159:      uint256 liquidity0;
160:      (, , liquidity0) = IUniswapV2Router02(uniswapRouter).addLiquidity(
161:        p12,
162:        gameCoinAddress,
163:        amountP12,
164:        amountGameCoinDesired,
165:        amountP12,
166:        amountGameCoinDesired,
167:        address(p12mine),
168:        getBlockTimestamp() + addLiquidityEffectiveTime
169:      );
```

Description

Based on the UniSwap API doc https://docs.uniswap.org/protocol/V2/reference/smart-contracts/router-02#addliquidity the `deadline` parameter is "*Unix timestamp after which the transaction will revert*". Because the transaction time baseline can shift overtime, a `deadline` value considered reasonable during deployment may become unusable at all in the future due to blockchain congestion or UniSwap transaction congestion. When that happens, `IUniswapV2Router02.addLiquidity()` call will likely to revert and cause `create()` function to fail as well.

Recommendation

Add a setter with `onlyOwner` modifier to set `addLiquidityEffectiveTime` parameter.

Client Response

Acknowledged. As long as `addLiquidityEffectiveTime` parameter is greater than 0 the transaction will not revert.

P12-5: `P12V0FactoryUpgradeable` P12 token approved `uniswapRouter` with `uint256` max amount

Category	Severity	Code Reference	Status
Privilege Related	Informational	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:54	Acknowledged

Code

```
54:     IERC20(p12).approve(uniswapRouter, type(uint256).max);
```

Description

We understand that approve “unlimited” allowance for Uniswap at once is commonly used to save gas so that the P12 token amount does not have to be approved every time on the fly when need it, also it is unlikely that Uniswap were to be compromised due to access or new bug discovery. However this opens the super privilege for an external contract account to transfer unlimited amount from P12 contract. Also, in case the P12 approved allowance for Uniswap is used up, there needs to be a way to “refill” the approved amount.

Recommendation

Add a function with `onlyOwner` modifier to update the approved amount in `uniswapRouter` for P12 amount.

Client Response

Acknowledged. The probability that we use up `type(uint256).max = 2^256-1` approved amount is zero. We also favor decentralization over centralized privilege hence will not add a function to adjust approved amount after deployment.

P12-6: `P12V0FactoryUpgradeable` mixed use of SafeMath and arithmetic operators hurts readability

Category	Severity	Code Reference	Status
Code Style	Informational	contracts/contracts/factory/P12V0FactoryUpgradeable.sol:98	Fixed

Code

```
    time = amountGameCoin.mul(delayK).div(P12V0ERC20(gameCoinAddress).totalSupply()) + 4 * delayB;
```

Description

SafeMath is generally not needed starting with Solidity 0.8.0, since the compiler now has built in overflow checking with a little more gas. The combination of SafeMath and normal arithmetic operators is not the best practice.

Recommendation

Use either SafeMath or normal arithmetic operators.

Client Response

Fixed.

P12-7: `P12V0FactoryUpgradeable::register()` developer address is not checked

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:127-130	Fixed

Code

```
127:    function register(string memory gameId, address developer) external virtual override
onlyOwner {
128:        allGames[gameId] = developer;
129:        emit RegisterGame(gameId, developer);
130:    }
```

Description

The input parameter `developer` address needs to be validated before registering it.

Recommendation

Validate `developer` is not `address(0)` .

Client Response

Fixed.

P12-8: `P12V0FactoryUpgradeable` p12mine can be used before assign a value

Category	Severity	Code Reference	Status
Logical Issue	Medium	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:159-169	Fixed

Code

```
    (, , liquidity0) = IUniswapV2Router02(uniswapRouter).addLiquidity(
      p12,
      gameCoinAddress,
      amountP12,
      amountGameCoinDesired,
      amountP12,
      amountGameCoinDesired,
      address(p12mine),
      getBlockTimestamp() + addLiquidityEffectiveTime
    );
```

Description

`p12mine` can be only set by the function `setP12Mine()` function. However, if `create()` function is called BEFORE `setP12Mine()`, `p12mine` is a zero address and the liquidity will be added to a zero address. This will potentially lead to loss of fund.

Recommendation

Validate `p12mine` is not `address(0)` in the `create()` function or pass `p12mine` as an input parameter in the `initialize()` function.

Client Response

Fixed.

P12-9: `P12V0FactoryUpgradeable::executeMint()` issue with non-existent `gameCoinAddress` and `mintId`

Category	Severity	Code Reference	Status
Logical Issue	Critical	Contracts/contracts/factory/P12V0FactoryUpgradeable.sol:248	Fixed

Code

```
247:     // check if it has been executed
248:     require(!coinMintRecords[gameCoinAddress][mintId].executed, 'this mint
has been executed');
```

Description

When a non-existent `gameCoinAddress` or `mintId` is passed to the `executeMint()` function, the `!coinMintRecords[gameCoinAddress][mintId].executed` will be `false` and `!false` will pass the require.

Recommendation

Add a require statement to require `coinMintRecords[gameCoinAddress][mintId].unlockTimestamp != 0`.

Client Response

Fixed.

P12-10: `P12V0FactoryUpgradeable::declareMintCoin()` logical issue with `mintId`

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/factory/ P12V0FactoryUpgradeable.sol:225	Acknowledged

Code

```
225:      bytes32 mintId = _hashOperation(gameCoinAddress, msg.sender, amountGameCoin, time,
_initHash);
```

Description

Assume the developer called `declareMintCoin()` for the first time at timestamp A. Note that the `coinMintRecords[gameCoinAddress][_preMintId].unlockTimestamp` is A + delayD where delayD is 10 minutes. One minute later developer calls it for the second time, and then the `mintId` will still be the same because the `time` is unchanged. In this scenario, `coinMintRecords[gameCoinAddress][mintId]` will be overwritten. We are not sure if this is the intended business logic.

Recommendation

Need P12 team confirm this is an intended business feature.

Client Response

Acknowledged. the `_hashOperation` function will compute the `mintId` based on previous value and this is intended.

P12-11: `Pair.salt` is not used

Category	Severity	Code Reference	Status
Code Style	Informational	contracts/contracts/secretShop/ERC1155Delegate.sol:21	Acknowledged

Code

```
20:   struct Pair {
21:     uint256 salt;
22:     IERC1155 token;
23:     uint256 tokenId;
24:     uint256 amount;
25:   }
```

Description

The `salt` in struct `Pair` is defined but not used anywhere in the code.

Recommendation

Identify if it is needed, if not needed, delete it; otherwise add corresponding code that uses it.

Client Response

Acknowledged. The `salt` will be generated and used off-chain.

P12-12: `Op.COMPLETE_BUY_OFFER` is not used

Category	Severity	Code Reference	Status
Code Style	Informational	contracts/contracts/secretShop/ MarketConsts.sol:110	Fixed

Code

```
    enum Op {
      INVALID,
      COMPLETE_SELL_OFFER,
      COMPLETE_BUY_OFFER,
      CANCEL_OFFER
    }
```

Description

The `COMPLETE_BUY_OFFER` in `Op` enum is defined but not used anywhere in the code. We understand `Op.INVALID` value default to 0 is to avoid the default non-existent 0 value collision for other states.

Recommendation

If there is no need for `COMPLETE_BUY_OFFER` enum, remove it. Or it should be checked as a condition in the `SecretShopUpgradable::_run()` function.

Client Response

Fixed.

P12-13: `SettleShared.amountToEth` is not used

Category	Severity	Code Reference	Status
Code Style	Informational	contracts/contracts/secretShop/MarketConsts.sol:81	Fixed

Code

```
  struct SettleShared {
    uint256 salt;
    uint256 deadline;
    uint256 amountToEth;
    address user;
    /**
     * can one order fail
     * if true, tx will revert if one order fail
     * else, tx didn't fail if one
     */
    bool canFail;
  }
```

Description

The `amountToEth` in `SettleShared` struct is defined but not used anywhere in the code.

Recommendation

Identify if this item is needed. If not, remove it; otherwise add the corresponding code which uses it.

Client Response

Fixed.

P12-14: Unused local variable `amountETH` in `SecretShopUpgradable`

Category	Severity	Code Reference	Status
Code Style	Informational	contracts/contracts/secretShop/ SecretShopUpgradable.sol:230-254	Fixed

Code

```
    for (uint256 i = 0; i < input.details.length; i++) {
      Market.SettleDetail memory detail = input.details[i];
      Market.Order memory order = input.orders[detail.orderIdx];
      if (input.shared.canFail) {
        try ISecretShopUpgradable(address(this)).runSingle(order, input.shared,
detail) returns (uint256 ethPayment) {
          amountEth -= ethPayment;
        } catch Error(string memory err) {
          emit EvFailure(i, bytes(err));
        } catch (bytes memory err) {
          emit EvFailure(i, err);
        }
      } else {
        amountEth -= _run(order, input.shared, detail);
      }
    }
```

Description

In the `run()` function, the local variable `amountEth` has been assigned value by `amountEth -= ethPayment` and `amountEth -= _run(order, input.shared, detail)`, but `amountEth` is never used or returned afterwards.

Recommendation

Update the logic to use the updated `amountEth` value.

Client Response

Fixed.

P12-15: `P12Asset::mint()` logical issue when `id` does not exist

Category	Severity	Code Reference	Status
Logical Issue	Medium	contracts/contracts/sft-factory/ P12Asset.sol:59-68	Fixed

Code

```
65:     require(amount + supply[id] <= maxSupply[id], 'P12Asset: exceed max supply');
66:     _mint(to, id, amount, data);
```

Description

When `id` does not exist and `amount` equals 0, it will pass the `require` statement and call `ERC1155::_mint()` in line 66. `ERC1155::_mint()` creates `amount` tokens of token type `id`, and assigns them to the `to` address. In this case, `id` token will be created with 0 amount.
We understand this function can only be called by the owner as there is `onlyOwner` modifier. However, it is best practice to be defensive.

Recommendation

Add a `require(id < idx, 'P12Asset: id is not valid’)` check before calling `_mint()`.

Client Response

Fixed.

P12-16: `P12Asset::_setUri()` logical issue with empty string

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/sft-factory/P12Asset.sol:95-99	Fixed

Code

```
  function _setUri(uint256 id, string calldata newUri) private {
    require(bytes(_uri[id]).length == 0, 'P12Asset: uri already set');
    _uri[id] = newUri;
    emit SetUri(id, newUri);
  }
```

Description

When `newUri` is an empty string, `require` statement will pass in the next call as the length is 0. Granted there is no harm in this function along that you can reassign a URI value when it is empty string, other functions use `_uri[id]` will return an invalid URI.

Recommendation

Validate `newUri` parameter before assigning it, check length > 0 at the very minimum.

Client Response

Fixed. There are changes on the logic and the new version allows update of `uri` and added the validation logic of empty string.

P12-17: `P12Asset::mint()` reentrancy risk to bypass `maxSupply`

Category	Severity	Code Reference	Status
Logical Issue	Critical	contracts/contracts/sft-factory/ P12Asset.sol:66,67	Fixed

Code

```
59:    function mint(
60:      address to,
61:      uint256 id,
62:      uint256 amount,
63:      bytes memory data
64:    ) public override onlyOwner {
65:      require(amount + supply[id] <= maxSupply[id], 'P12Asset: exceed max supply');
66:      _mint(to, id, amount, data);
67:      supply[id] += amount;
68:    }
```

Description

When `_mint()` is called, the `IERC1155Receiver(to).onERC1155Received()` will be called internally with the `to` address, which makes it possible for the external `to` contract to call `mint()` again. Since `supply[id]` is only updated after the `_mint()` call, the reentrant call will still pass the `maxSupply` check. Granted the `mint()` function is with `onlyOwner` modifier which means only the P12Asset creator (game developer) can call it, there is still economical incentives for them to to bypass the `maxSupply` check later to mint more P12Asset than what is configured initially, especially when that game asset price is high. Hence it's best to be defensive from P12 platform perspective.

Recommendation

Follow the Checks-Effects-Interactions pattern and update the `supply[id] += amount` before calling `_mint()` function.

Client Response

Fixed.

P12-18: `P12AssetFactoryUpgradable::initialize()` does not validate address

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/sft-factory/P12AssetFactoryUpgradable.sol:37	Fixed

Code

```
  function initialize(address p12factory_) public initializer {
    p12factory = p12factory_;
```

Description

The input parameter `p12factory_` input address could be zero.

Recommendation

Validate `p12factory_` is not `address(0)`.

Client Response

Fixed.

P12-19: No `contractURI` validation in `P12AssetFactoryUpgradable::createCollection()`

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/sft-factory/ P12AssetFactoryUpgradable.sol:59-70	Fixed

Code

```
    P12Asset collection = new P12Asset(contractURI);
```

Description

The `createCollection()` function does not validate the input `contractURI`. It directly creates a new P12Asset without validating `contractURI`. Also, the `P12Asset::constructor()` does not check it neither.

Recommendation

Validate `contractURI` is not empty string.

Client Response

Fixed.

P12-20: `P12MineUpgradeable::initialize()` does not validate address

Category	Severity	Code Reference	Status
Logical Issue	Informational	contracts/contracts/staking/P12MineUpgradeable.sol:50	Fixed

Code

```
  function initialize(
    address p12Token_,
    address p12Factory_,
    uint256 startBlock_,
    uint256 delayK_,
    uint256 delayB_
  ) public initializer {
    p12Token = p12Token_;
    p12Factory = p12Factory_;
    p12RewardVault = address(new P12RewardVault(p12Token_));
    startBlock = startBlock_;
    delayK = delayK_;
    delayB = delayB_;
```

Description

The input parameter `p12Token_` and `p12Factory_` address can be zero. And `startBlock_` should be greater than some reasonable `block.number` value. For example 14921591 for Ethereum. And `delayK_` and `delayB_` based on business requirement.

Recommendation

Add `require` statement to validate the input parameters.

Client Response

Fixed. As the final blockchain has not been determined, the value of `startBlock_` cannot be determined yet.

P12-21: `P12RewardVault::constructor()` does not validate address

Category	Severity	Code Reference	Status
Logical Issue	Informational	Contracts/contracts/staking/ P12RewardVault.sol:17	Fixed

Code

```
constructor(address p12Token_) {
    p12Token = p12Token_;
}
```

Description

The input parameter `p12Token_` address can be zero.

Recommendation

Add `require` statement to validate the input parameter.

Client Response

Fixed.

P12-22: `P12Token` unlimited mint

Category	Severity	Code Reference	Status
Privilege Related	Medium	Contracts/contracts/token/P12Token.sol: 21-23	Acknowledged

Code

```
function mint(address recipient, uint256 amount) public override onlyOwner {
  _mint(recipient, amount);
}
```

Description

The `P12Token` does not have a total supply defined in the contract as the `totalSupply` is a parameter passed into the constructor. Also owner can mint more tokens via the `mint` function any time after the deployment.

Recommendation

A predefined `totalSupply` constant number in the contract would be preferred for a more transparent and stable tokenomics. Also consider remove the `mint` function so that owner does not have super privilege to mint unlimited tokens.

Client Response

Acknowledged. The project are still designing the tokenomics and will further make changes to the P12Token mint logic once the model is determined and published in the white paper.

Disclaimer