Unblock Labs

An EatTheBlocks Company

Audit report

FireBot - FireVaultFBX

October 2022

Table of Contents

Summary

This report has been prepared by Unblock Labs for FireBot to discover issues and vulnerabilities in the source code of their FireVaultFBXV2 and ElementalParticles smart contracts as well as any contract dependencies used in the project. A comprehensive examination has been performed utilizing Static Analysis and Manual Code Review techniques

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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project summary

Project name	FireBot
Platform	Polygon
Language	Solidity

Audit summary

Delivery date	October 27, 2022
Methodology	Static Analysis, Manual Review

Vulnerability summary

Level	Total	Acknowledge	Mitigated	Resolved
● **Critical**	1	0	0	1
● **High**	2	0	0	2
● **Medium**	0	0	0	0
● **Low**	9	0	0	9
● **Information**	0	0	0	0
● **Discussion**	0	0	0	0

Audit scope

ID	Contract	Codebase
EP	ElementalParticles.sol	https://polygonscan.com/address/0xFb0F33679639d7BfC9cfb80d4eE519F21552F504
FV	FireVaultFBXV2.sol	https://polygonscan.com/address/0xf584be26441bf224a91d4f6bb0320b7c9f4ef875

Revised Codebase	
ElementalParticles.sol	https://polygonscan.com/address/0x898fa6c1436a0c7514bd2215405591e71e665234
FireVaultFBXV2.sol	https://polygonscan.com/address/0xda6167d718b7439b8eca16e011d2d85c2c7046d1

Findings

ID	Title	Category	Severity	Status
EP-01	Tokens are not transferred back when unstaking	Volatile Code	🔴 Critical	Resolved
EP-02	Incorrect “transferFrom” in stakeEP	Volatile Code	🟠 High	Resolved
FV-01	Invalid “approve” in claimRewardsAndBalance	Volatile Code	🟠 High	Resolved
EP-03	Potential revert in daily FBX Emission PeEP	Volatile Code	🟡 Low	Resolved
EP-04	No events emitted during staking and unstaking	Coding style	🟡 Low	Resolved
EP-05	Duplicate variables	Gas optimisation	🟡 Low	Resolved
EP-06	Storage gas optimisation	Gas optimisation	🟡 Low	Resolved
FV-02	No events emitted during deposit and withdraw	Coding style	🟡 Low	Resolved
FV-03	Missing input validation	Volatile Code	🟡 Low	Resolved
FV-04	Duplicate variables	Gas optimisation	🟡 Low	Resolved
FV-05	Use of ERC20Burnable	Volatile Code	🟡 Low	Resolved
FV-06	No added value in swapFBXforFireFBX and swapFireFBXForFBX	Coding style	🟡 Low	Resolved

EP-01 | Tokens are not transferred back when unstaking

Category	Severity	Location	Status
Volatile Code	● Critical	ElementalParticles.sol: 1076	Resolved

Description

The function `unstakeEP()` uses the function `transfer()` to send the staked tokens back to the user.
Since this function is executed within the ERC20's contract of the transferred token, this call actually transfers tokens from `msg.sender` to his own wallet and does not transfer the staked tokens back from the contract.

```
function unstakeEP(uint256 amountEP) public {
  ...
  transfer(msg.sender, amountEP);
}
```

In @openzeppelin/contracts/token/ERC20/ERC20.sol:113~117:

```
function transfer(address to, uint256 amount) public virtual override
returns (bool) {
  address owner = _msgSender();
  _transfer(owner, to, amount);
  return true;
}
```

Recommendation

Use the internal function `_transfer()` directly.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

EP-02 | Incorrect “transferFrom” in stakeEP

Category	Severity	Location	Status
Volatile Code	● High	ElementalParticles.sol: 1067	Resolved

Description

The function `stakeEP()` uses the function `transferFrom()` to transfer the tokens to stake from the user.
Since this function is executed within the ERC20’s contract of the transferred token, this call actually requires msg.sender to approve his own spending.

```
function stakeEP(uint256 amountEP) public {
  …
  transferFrom(msg.sender, address(this), amountEP);
}
```

In @openzeppelin/contracts/token/ERC20/ERC20.sol:158~167

```
function transferFrom(
address from,
address to,
uint256 amount
) public virtual override returns (bool) {
  address spender = _msgSender();
  _spendAllowance(from, spender, amount);
  _transfer(from, to, amount);
  return true;
}
```

Recommendation

Use the function `transfer()`.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

EP-03 | Potential revert in dailyFBXEmissionPerEP

Category	Severity	Location	Status
Volatile Code	● Low	ElementalParticles.sol: 1034	Resolved

Description

The function `dailyFBXEmissionPerEP()` will perform a division by **0** and revert if the balance of the contract is **0**.

```
function dailyFBXEmissionPerEP() public view returns(uint256) {
  return 1e18 * dailyFBXEmission() / balanceOf(address(this));
}
```

Recommendation

Handle the case appropriately to return **0** when the balance is empty.

Alleviation

`[UnblockLabs]`: The client opted to remove this code

EP-04 | No events emitted during staking and unstaking

Category	Severity	Location	Status
Coding style	🟡 Low	ElementalParticles.sol: 1064~1069; 1071~1077;	Resolved

Description

The following functions do not emit events to pass the changes out of chain.

- `stakeEP()`
- `unstakeEP()`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that changed during runtime.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

EP-05 | Duplicate variables

Category	Severity	Location	Status
Gas optimisation	🟡 Low	ElementalParticles.sol: 994; 995;	Resolved

Description

The following 2 properties stores the same value and should be merged into one:

- `SAFE_FBX`
- `FBX`

Recommendation

Change the interface `IFBX` to extend `IERC20` and use only 1 variable.

```
interface IFBX is IERC20 {
  function burnFrom(address account, uint256 amount) external;
}
contract ElementalParticles is ERC20, ERC20Burnable {
  IFBX public constant FBX =
IFBX(0xD125443F38A69d776177c2B9c041f462936F8218);
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

EP-06 | Storage gas optimisation

Category	Severity	Location	Status
Gas optimisation	● Low	ElementalParticles.sol: 1000~1002;	Resolved

Description

The following 3 mapping variables stores informations with the same key and are used together within the implementation of the contract:

- `mapping(address => uint256) public stakedEP`
- `mapping(address => uint256) public lastClaim`
- `mapping(address => uint256) public amountClaimed`

Recommendation

Grouping the informations in a specific struct can help improve the gas used:

```
struct UserInfo {
  uint256 stakedEP;
  uint256 lastClaim;
  uint256 amountClaimed;
}

mapping(address => uint256) public userInfos;
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

FV-01 | Invalid “approve” in claimRewardsAndBalance

Category	Severity	Location	Status
Volatile Code	● High	FireVaultFBXV2.sol: 1624	Resolved

Description

The function `claimRewardsAndBalance()` approves itself as a spender before staking its own tokens.
This is linked to the #FV-02 issue.

```
// Stake unstaked EP
uint256 unstakedEP = SAFE_EP.balanceOf(address(this));
if (unstakedEP > 0) {
  SAFE_EP.approve(address(this), unstakedEP);
   EP.stakeEP(unstakedEP);
}
```

Recommendation

Fix the issue #FV-02 and update the code to approve the `EP` contract as the spender.

Alleviation

`[UnblockLabs]`: The client opted to remove the approval necessity when staking.

FV-02 | No events emitted during deposit and withdraw

Category	Severity	Location	Status

Coding style	● Low	FireVaultFBXV2.sol: 1633~1637;
		1639~1643;
			Resolved

Description

The following functions do not emit events to pass the changes out of chain.

- `deposit()`
- `withdraw()`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are changed during runtime.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

FV-03 | Missing input validation

Category	Severity	Location	Status

Volatile code	● Low	FireVaultFBXV2.sol: 1633~1637;
		1639~1643;
			Resolved

Description

The functions `deposit()` and `withdraw()` do not validate that the amount sent in the parameter is greater than **0**.

Recommendation

Validate the input parameters passed to the functions.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

FV-04 | Duplicate variables

Category	Severity	Location	Status

Gas optimisation	🟡 Low	FireVaultFBXV2.sol: 1554~1566;
		1570~1573;
			Resolved

Description

The following properties stores the same value and should be merged:

- `SUSHI_ROUTER_ADDRESS` / `SUSHI_ROUTER`
- `FBX_CONTRACT_ADDRESS` / `SAFE_FBX`
- `EP_CONTRACT_ADDRESS` / `SAFE_EP` / `EP`

Recommendation

Change the interface `IEP` to extend `IERC20` and use only 1 variable.

```
interface IEP is IERC20 {
  …
}
contract FireVaultFBXV2 is ERC20, ERC20Burnable, ERC20Permit {
  using SafeERC20 for IERC20;

  IERC20 public constant FBX =
IERC20(0xD125443F38A69d776177c2B9c041f462936F8218);
  IEP public constant EP =
IEP(0xF581bd6418603C2754701Ff80FB1EA983d7767AB);
  ISushiRouter public constant ROUTER =
ISushiRouter(0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506);
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

FV-05 | Use of ERC20Burnable

Category	Severity	Location	Status
Volatile code	● Low	FireVaultFBXV2.sol: 1562;	Resolved

Description

The contract inherits `ERC20Burnable` which makes the function `burn()` available publicly.
A user can call this function directly and lose the `FBX` that he was untitled if calling the `withdraw()` function.

Recommendation

If this feature is not required, we recommend removing the inheritance to `ERC20Burnable` and implement a specific `burnFrom` function to be able to burn `FBX` from the `ElementalParticles` contract.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes

FV-06 | No added value in swapFBXforFireFBX and swapFireFBXForFBX

Category	Severity	Location	Status
Coding style	● Low	FireVaultFBXV2.sol: 1645~1647; 1649~1651;	Resolved

Description

The functions `swapFBXForFireFBX()` and `swapFireFBXForFBX()` do not add any value over the functions `deposit()` and `withdraw()` as they only redirect the call.

```
function swapFBXforFireFBX(uint256 amountFBX) public {
  deposit(amountFBX);
}

function swapFireFBXForFBX(uint256 amountFireFBX) public {
  withdraw(amountFireFBX);
}
```

Recommendation

To improve maintainability and simplicity of the code, we suggest removing the functions and declaring `deposit()` and `withdraw()` as `external` functions.

Alleviation

`[UnblockLabs]`: The client opted to remove those functions.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Unblock Labs's prior written consent in each instance.

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 Unblock Labs 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 intended 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. Unblock Labs's position is that each company and individual are responsible for their own due diligence and continuous security. Unblock Labs'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 Unblock Labs are subject to dependencies and 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.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, UNBLOCK LABS HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, UNBLOCK LABS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER UNBLOCK LABS NOR ANY OF UNBLOCK LABS’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. UNBLOCK LABS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT UNBLOCK LABS’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.
THE REPRESENTATIONS AND WARRANTIES OF UNBLOCK LABS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Unblock Labs

An EatTheBlocks Company

www.unblock-labs.com