



# Venus Token Converter

## AUDIT REPORT

Version 1.0.0

Serial No. 2023082800012028

Presented by Fairyproof

August 28, 2023

# 01. Introduction

---

This document includes the results of the audit performed by the Fairyproof team on the Venus TokenConverter project.

**Audit Start Time:**

August 20, 2023

**Audit End Time:**

August 25, 2023

**Audited Source File's Address:**

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/AbstractTokenConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/IAbstractTokenConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/RiskFundConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/XVSVaultConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/Utils/Constants.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/Utils/Validators.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/RiskFundStorage.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/RiskFundV2.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/XVSVaultTreasury.sol>

**Audited Code's Github Repository:**

<https://github.com/VenusProtocol/protocol-reserve/>

**Audited Code's Github Commit Number When Audit Started:**

b6ba4393f70a765dda54c399f0a57ade137296af

**Audited Code's Github Commit Number When Audit Ended:**

0fb7f1d6e00723baee6213f2840e0f097137a89a

**Audited Source Files:**

The source files audited include all the files as follows:

```

├─ ProtocolReserve
│   └─ RiskFundStorage.sol
│   └─ RiskFundV2.sol
│   └─ xVSVaultTreasury.sol
├─ TokenConverter
│   └─ AbstractTokenConverter.sol
│   └─ IAbstractTokenConverter.sol
│   └─ RiskFundConverter.sol
│   └─ xVSVaultConverter.sol
└─ Utils
    └─ Constants.sol
    └─ Validators.sol

```

The goal of this audit is to review Venus's solidity implementation for its TokenConverter function, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the Venus team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

## — Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on

any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

## — Methodology

---

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairproof auditing process follows a routine series of steps:

### 1. Code Review, Including:

- Project Diagnosis

Understanding the size, scope and functionality of your project's source code based on the specifications, sources, and instructions provided to Fairproof.

- Manual Code Review

Reading your source code line-by-line to identify potential vulnerabilities.

- Specification Comparison

Determining whether your project's code successfully and efficiently accomplishes or executes its functions according to the specifications, sources, and instructions provided to Fairproof.

### 2. Testing and Automated Analysis, Including:

- Test Coverage Analysis

Determining whether the test cases cover your code and how much of your code is exercised or executed when test cases are run.

- Symbolic Execution

Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.

### 3. Best Practices Review

Reviewing the source code to improve maintainability, security, and control based on the latest established industry and academic practices, recommendations, and research.

## — Structure of the document

---

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

## — Documentation

For this audit, we used the following source(s) of truth about how the token issuance function should work:

Website: <https://venus.io/>

Whitepaper: <https://github.com/VenusProtocol/venus-protocol-documentation/tree/main/whitepapers>

Source Code:

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/AbstractTokenConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/IAbstractTokenConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/RiskFundConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/TokenConverter/XVSVaultConverter.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/Utils/Constants.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/Utils/Validators.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/RiskFundStorage.sol>

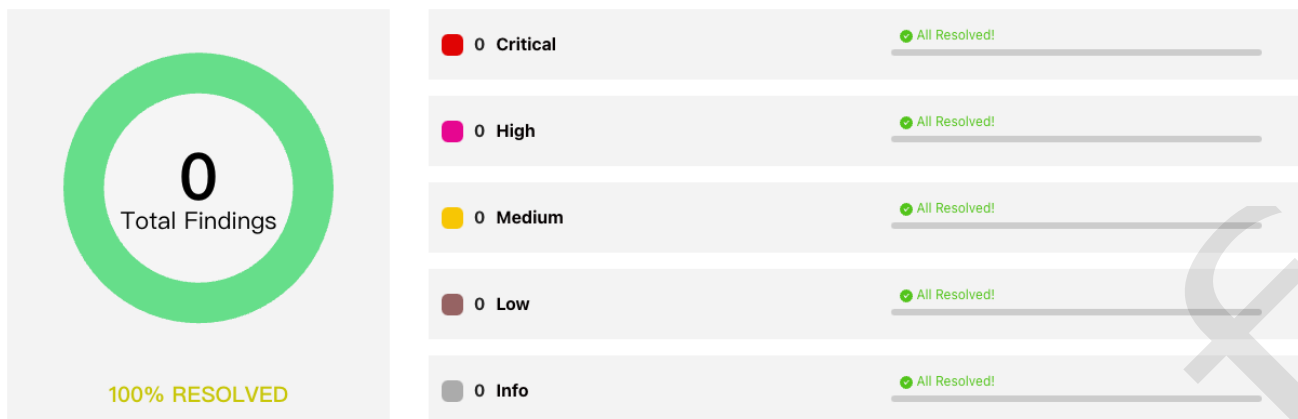
<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/RiskFundV2.sol>

<https://github.com/VenusProtocol/protocol-reserve/blob/feat/passive-swap/contracts/ProtocolReserve/XVSVaultTreasury.sol>

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Venus team or reported an issue.

## — Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2023082800012028	Fairyproof Security Team	Aug 20, 2023 - Aug 25, 2023	Passed



#### Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, no issues were uncovered.

## 02. About Fairyproof

[Fairyproof](#) is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

## 03. Introduction to Venus

Venus Protocol generates incomes in different underlying tokens. These tokens will be transferred to the ProtocolShareReserve contract, and from that contract, the income is distributed to different targets (RiskFundConverter, VTreasury, Prime, XSVVaultConverter), following the updated Tokenomics. XSVVaultConverter and RiskFundConverter need to convert the received income into specific tokens (XVS in the XSVVaultConverter and USDT in the RiskFundConverter). Every time a conversion is completed, the received tokens are sent to the XSVVaultTreasury and to the RiskFund contracts respectively.

The above description is quoted from relevant documents of Venus.

## 04. Major functions of audited code

The audited code mainly implements a token conversion function which incentivizes users to exchange any token to a specific token. The conversion rules are made based on the rate of the two token prices fed by an oracle.

## 05. Coverage of issues

---

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Access Control
- Admin Rights
- Arithmetic Precision
- Code Improvement
- Contract Upgrade/Migration
- Delete Trap
- Design Vulnerability
- DoS Attack
- EOA Call Trap
- Fake Deposit
- Function Visibility
- Gas Consumption
- Implementation Vulnerability
- Inappropriate Callback Function
- Injection Attack
- Integer Overflow/Underflow
- IsContract Trap
- Miner's Advantage
- Misc
- Price Manipulation
- Proxy selector clashing
- Pseudo Random Number
- Re-entrancy Attack
- Replay Attack
- Rollback Attack
- Shadow Variable
- Slot Conflict
- Token Issuance
- Tx.origin Authentication
- Uninitialized Storage Pointer

## 06. Severity level reference

---

Every issue in this report was assigned a severity level from the following:

**Critical** severity issues need to be fixed as soon as possible.

**High** severity issues will probably bring problems and should be fixed.

**Medium** severity issues could potentially bring problems and should eventually be fixed.

**Low** severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

**Informational** is not an issue or risk but a suggestion for code improvement.

## 07. Major areas that need attention

---

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

### - Function Implementation

---

We checked whether or not the functions were correctly implemented.  
We didn't find issues or risks in these functions or areas at the time of writing.

### - Access Control

---

We checked each of the functions that could modify a state, especially those functions that could only be accessed by owner or administrator  
We didn't find issues or risks in these functions or areas at the time of writing.

### - Token Issuance & Transfer

---



We examined token issuance and transfers for situations that could harm the interests of holders.  
We didn't find issues or risks in these functions or areas at the time of writing.

## **- State Update**

---

We checked some key state variables which should only be set at initialization.  
We didn't find issues or risks in these functions or areas at the time of writing.

## **- Asset Security**

---

We checked whether or not all the functions that transfer assets were safely handled.  
We didn't find issues or risks in these functions or areas at the time of writing.

## **- Miscellaneous**

---

We checked the code for optimization and robustness.  
We didn't find issues or risks in these functions or areas at the time of writing.

## **08. issues by severity**

---

**- N/A**

---

## **09. Issue descriptions**

---

**- N/A**

---

## **10. Recommendations to enhance the overall security**

---

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- N/A

## 11. Appendices

### 11.1 Unit Test

#### 1. XVSVaultConverter.t.js

```
const {time, loadFixture, impersonateAccount} = require("@omicfoundation/hardhat-network-helpers");
const { expect } = require("chai");
const { ethers } = require("hardhat");

describe("XVSTokenConverter Unit Test", function () {
  const ZERO_ADDR = ethers.constants.AddressZero
  const WBNB_ADDR = "0xbb4CdB9CBd36B01bd1cBaEBF2De08d9173bc095c"
  const VBNB_ADDR = "0xA07c5b74C9B40447a954e1466938b865b6BBEa36"
  const MANTISSA_ONE = ethers.utils.parseUnits('1', '18')
  const EXP_SCALE = ethers.utils.parseUnits('1', '18')

  const XVS_VAULT = "0x051100480289e704d20e9DB4804837068f3f9204"
  const XVS_TOKEN = "0xcF6BB5389c92Bdda8a3747Ddb454cB7a64626c63"
  const PRICE_ORACLE = "0x6592b5DE802159F3E74B2486b091D11a8256ab8A"

  async function deployXVSVaultTreasury(_owner) {
    const XVSVaultTreasuryFactory = await ethers.getContractFactory('XVSVaultTreasury',
      _owner)
    const XVSVaultTreasuryImp = await XVSVaultTreasuryFactory.deploy(XVS_TOKEN)
    const BeaconFactory = await
ethers.getContractFactory('@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol:Upgra
deableBeacon', _owner)
    const Beacon = await BeaconFactory.deploy(XVSVaultTreasuryImp.address)
    const BeaconProxyFactory = await
ethers.getContractFactory('@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol:BeaconProxy
', _owner)
    const BeaconProxy = await BeaconProxyFactory.deploy(Beacon.address, '0x')
    const XVSVaultTreasury = XVSVaultTreasuryFactory.attach(BeaconProxy.address)
    return XVSVaultTreasury
  }
}
```

```

async function deployAndBindFixture() {
  // get users;
  const [owner, Alice, Bob, ...users] = await ethers.getSigners()

  const impersonateAccountXVS = "0xF977814e90dA44bFA03b6295A0616a897441aceC"
  await impersonateAccount(impersonateAccountXVS);
  const signerXVS = await ethers.getSigner(impersonateAccountXVS)

  const XVSVaultTreasury = await deployXVSVaultTreasury(owner)

  const ACMFactory = await ethers.getContractFactory('MockACM', owner)
  const ACM = await ACMFactory.deploy()

  const XVSVaultConverterFactory = await
ethers.getContractFactory('XVSVaultConverter', owner)
  const XVSVaultConverter = await XVSVaultConverterFactory.deploy()

  await XVSVaultTreasury.initialize(ACM.address, XVS_VAULT)
  await XVSVaultConverter.initialize(ACM.address, PRICE_ORACLE,
XVSVaultTreasury.address)

  const XVSVault = await ethers.getContractAt("IXSVault", XVS_VAULT)
  const PriceOracle = await
ethers.getContractAt("@venusprotocol/oracle/contracts/ResilientOracle.sol:ResilientOracle",
PRICE_ORACLE)
  const wbnb = await
ethers.getContractAt("@openzeppelin/contracts/token/ERC20/IERC20.sol:IERC20", WBNB_ADDR)
  const xvs = await
ethers.getContractAt("@openzeppelin/contracts/token/ERC20/IERC20.sol:IERC20", XVS_TOKEN)

  const DeflatingTokenFactory = await ethers.getContractFactory('MockDeflatingToken',
owner)
  const deflatingToken = await
DeflatingTokenFactory.deploy(ethers.utils.parseEther('10000'))

  await owner.sendTransaction({value: ethers.utils.parseEther('10'), to:
wbnb.address})

  // return
  return { owner, Alice, Bob, signerXVS, XVSVaultTreasury, XVSVaultConverter,
PriceOracle, ACM, wbnb, xvs, deflatingToken, XSVVault };
}

describe("XVSTokenConverter init unit test", function() {
  it("Initial state should equal with the params of constructor", async function() {
    const { XVSVaultTreasury, XVSVaultConverter } = await
loadFixture(deployAndBindFixture)
    expect(await XVSVaultTreasury.xvsAddress()).to.be.equal(XVS_TOKEN)
    expect(await XVSVaultTreasury.xsvVault()).to.be.equal(XVS_VAULT)
    expect(await XVSVaultConverter.priceOracle()).to.be.equal(PRICE_ORACLE)
    expect(await
XVSVaultConverter.destinationAddress()).to.be.equal(XSVVaultTreasury.address)
    expect(await XVSVaultConverter.conversionPaused()).to.be.equal(false)
  })
})

```

```

    });
  });

  describe("ACM and OnlyOwner unit test", function() {
    it("OnlyOwner unit test", async function() {
      const { Alice, XSVVaultTreasury, XSVVaultConverter } = await
loadFixture(deployAndBindFixture)
      await
expect(XSVVaultTreasury.connect(Alice).setXSVVault(XVS_VAULT)).to.be.revertedWith("Ownable:
caller is not the owner")
      await
expect(XSVVaultConverter.connect(Alice).setPriceOracle(PRICE_ORACLE)).to.be.revertedWith("O
wnable: caller is not the owner")
      await
expect(XSVVaultConverter.connect(Alice).setDestination(XSVVaultTreasury.address)).to.be.rev
ertedWith("Ownable: caller is not the owner")
      await expect(XSVVaultConverter.connect(Alice).sweepToken(WBNB_ADDR,
Alice.address, 10000)).to.be.revertedWith("Ownable: caller is not the owner")
    });

    it("ACM unit test", async function() {
      const { Alice, XSVVaultTreasury, XSVVaultConverter, ACM } = await
loadFixture(deployAndBindFixture)
      const functionSigFundXSVVault = "fundXSVVault(amountMantissa)"
      await
expect(XSVVaultTreasury.connect(Alice).fundXSVVault(10000)).to.be.revertedWithCustomError(
XSVVaultTreasury, "Unauthorized"
).withArgs(Alice.address, XSVVaultTreasury.address, functionSigFundXSVVault)

      const functionSigPauseConversion = "pauseConversion()"
      await
expect(XSVVaultConverter.connect(Alice).pauseConversion()).to.be.revertedWithCustomError(
XSVVaultConverter, "Unauthorized"
).withArgs(Alice.address, XSVVaultConverter.address,
functionSigPauseConversion)

      const functionSigResumeConversion = "resumeConversion()"
      await
expect(XSVVaultConverter.connect(Alice).resumeConversion()).to.be.revertedWithCustomError(
XSVVaultConverter, "Unauthorized"
).withArgs(Alice.address, XSVVaultConverter.address,
functionSigResumeConversion)

      const functionSigSetConversionConfig = "setConversionConfig(ConversionConfig)"
      await expect(XSVVaultConverter.connect(Alice).setConversionConfig([WBNB_ADDR,
VBNB_ADDR, ethers.utils.parseUnits('1', '18'), true])).to.be.revertedWithCustomError(
XSVVaultConverter, "Unauthorized"
).withArgs(Alice.address, XSVVaultConverter.address,
functionSigSetConversionConfig)
    });
  });

  describe("ConversionConfig unit test", function() {

```

```

    it("Set Zero Address should failed", async function() {
      const { owner, XSVVaultConverter, ACM } = await
loadFixture(deployAndBindFixture)
      await ACM.giveCallPermission(XSVVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
      const newConversionConfig = [
        ZERO_ADDR,
        ZERO_ADDR,
        ethers.utils.parseUnits('6', '18'),
        true
      ]
      await
expect(XSVVaultConverter.setConversionConfig(newConversionConfig)).to.be.revertedWithCustom
Error(
      XSVVaultConverter, "ZeroAddressNotAllowed"
    )
    });
    it("Set INCENTIVE too high should failed", async function() {
      const { owner, XSVVaultConverter, ACM } = await
loadFixture(deployAndBindFixture)
      await ACM.giveCallPermission(XSVVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
      const newConversionConfig = [
        WBNB_ADDR,
        XVS_TOKEN,
        ethers.utils.parseUnits('6', '18'),
        true
      ]
      await
expect(XSVVaultConverter.setConversionConfig(newConversionConfig)).to.be.revertedWithCustom
Error(
      XSVVaultConverter, "IncentiveTooHigh"
    ).withArgs(ethers.utils.parseUnits('6', '18'), ethers.utils.parseUnits('5',
'18'))
    });
  });

  describe("pauseConversion unit test", function() {
    it("Cannot convert tokens when paused", async function() {
      const { owner, XSVVaultConverter, ACM } = await
loadFixture(deployAndBindFixture)
      await ACM.giveCallPermission(XSVVaultConverter.address, "pauseConversion()",
owner.address)
      await XSVVaultConverter.pauseConversion()

      await expect(XSVVaultConverter.convertExactTokens(100, 10000, WBNB_ADDR,
XVS_TOKEN, owner.address)).to.be.revertedWithCustomError(
        XSVVaultConverter, "ConversionTokensPaused"
      )
    });
  });

  describe("sweepToken unit test", function() {

```

```

    it("sweepToken Zero Address should failed", async function() {
        const { owner, XSVVaultConverter, ACM } = await
loadFixture(deployAndBindFixture)

        await expect(XSVVaultConverter.sweepToken(ZERO_ADDR, ZERO_ADDR,
0)).to.be.revertedWithCustomError(
            XSVVaultConverter, "ZeroAddressNotAllowed"
        )
    });
});

describe("XSVVaultTreasury unit test", function() {
    it("fundXSVVault test", async function() {
        const { owner, signerXVS, XSVVaultTreasury, ACM, xvs, XSVVault } = await
loadFixture(deployAndBindFixture)

        const xvsStore = await XSVVault.xvsStore()
        const amountIn = ethers.utils.parseEther('1')
        await xvs.connect(signerXVS).transfer(XSVVaultTreasury.address, amountIn)
        await ACM.giveCallPermission(XSVVaultTreasury.address,
"fundXSVVault(amountMantissa)", owner.address)

        const xvsBalanceBefore = await xvs.balanceOf(xvsStore)
        await XSVVaultTreasury.fundXSVVault(amountIn)
        const xvsBalanceAfter = await xvs.balanceOf(xvsStore)

        expect(xvsBalanceAfter.sub(xvsBalanceBefore)).to.be.equal(amountIn)
    });
});

describe("convertTokens unit test", function() {
    it("convertExactTokens test", async function() {
        const { owner, signerXVS, XSVVaultTreasury, XSVVaultConverter, PriceOracle,
ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

        // set conversionConfig
        await ACM.giveCallPermission(XSVVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
        const incentive = ethers.utils.parseUnits('1', '18')
        const conversionConfig = [
            XVS_TOKEN,
            WBNB_ADDR,
            incentive,
            true
        ]
        await XSVVaultConverter.setConversionConfig(conversionConfig)

        // Preparation
        const amountIn = ethers.utils.parseEther('1')
        await wbnb.transfer(XSVVaultConverter.address, ethers.utils.parseEther('10'))
        await xvs.connect(signerXVS).transfer(owner.address, amountIn)
    });
});

```

```

    expect(await
wbnb.balanceOf(XVSVaultConverter.address)).to.be.equal(ethers.utils.parseEther('10'))
    expect(await xvs.balanceOf(owner.address)).to.be.equal(amountIn)

    // Get token price
    await PriceOracle.updateAssetPrice(XVS_TOKEN)
    const xvsPrice = await PriceOracle.getPrice(XVS_TOKEN)
    await PriceOracle.updateAssetPrice(WBNB_ADDR)
    const wbnbPrice = await PriceOracle.getPrice(WBNB_ADDR)

    await xvs.approve(XVSVaultConverter.address, amountIn)
    await XVSVaultConverter.convertExactTokens(
        amountIn,
        ethers.utils.parseEther('0'),
        XVS_TOKEN,
        WBNB_ADDR,
        owner.address
    )

    const amountOut =
xvsPrice.mul(MANTISSA_ONE.add(incentive)).div(wbnbPrice).mul(amountIn).div(EXP_SCALE)
    expect(await xvs.balanceOf(XVSVaultTreasury.address)).to.be.equal(amountIn)
    expect(await wbnb.balanceOf(owner.address)).to.be.equal(amountOut)
});

it("convertForExactTokens test", async function() {
    const { owner, signerXVS, XVSVaultTreasury, XVSVaultConverter, PriceOracle,
ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

    // set conversionConfig
    await ACM.giveCallPermission(XVSVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
    const incentive = ethers.utils.parseUnits('1', '18')
    const conversionConfig = [
        XVS_TOKEN,
        WBNB_ADDR,
        incentive,
        true
    ]
    await XVSVaultConverter.setConversionConfig(conversionConfig)

    // Preparation
    const amountOut = ethers.utils.parseEther('1')
    await wbnb.transfer(XVSVaultConverter.address, amountOut)
    await xvs.connect(signerXVS).transfer(owner.address,
ethers.utils.parseEther('1000'))

    expect(await wbnb.balanceOf(XVSVaultConverter.address)).to.be.equal(amountOut)
    expect(await
xvs.balanceOf(owner.address)).to.be.equal(ethers.utils.parseEther('1000'))

    // Get token price
    await PriceOracle.updateAssetPrice(XVS_TOKEN)

```

```

const xvsPrice = await PriceOracle.getPrice(XVS_TOKEN)
await PriceOracle.updateAssetPrice(WBNB_ADDR)
const wbnbPrice = await PriceOracle.getPrice(WBNB_ADDR)

const wbnbBefore = await wbnb.balanceOf(owner.address)

await xvs.approve(XSVVaultConverter.address, ethers.utils.parseEther('1000'))
await XSVVaultConverter.convertForExactTokens(
  ethers.utils.parseEther('1000'),
  amountOut,
  XVS_TOKEN,
  WBNB_ADDR,
  owner.address
)

const amountIn =
amountOut.mul(EXP_SCALE).div(xvsPrice.mul(MANTISSA_ONE.add(incentive))).div(wbnbPrice))
const wbnbAfter = await wbnb.balanceOf(owner.address)

expect(await xvs.balanceOf(XSVVaultTreasury.address)).to.be.equal(amountIn)
expect(wbnbAfter.sub(wbnbBefore)).to.be.equal(amountOut)
});

it("convertExactTokensSupportingFeeOnTransferTokens test", async function() {
  const { owner, signerXVS, XSVVaultTreasury, XSVVaultConverter, PriceOracle,
  ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

  // set conversionConfig
  await ACM.giveCallPermission(XSVVaultConverter.address,
  "setConversionConfig(ConversionConfig)", owner.address)
  const incentive = ethers.utils.parseUnits('1', '18')
  const conversionConfig = [
    XVS_TOKEN,
    WBNB_ADDR,
    incentive,
    true
  ]
  await XSVVaultConverter.setConversionConfig(conversionConfig)

  // Preparation
  const amountIn = ethers.utils.parseEther('1')
  await wbnb.transfer(XSVVaultConverter.address, ethers.utils.parseEther('10'))
  await xvs.connect(signerXVS).transfer(owner.address, amountIn)

  expect(await
wbnb.balanceOf(XSVVaultConverter.address)).to.be.equal(ethers.utils.parseEther('10'))
  expect(await xvs.balanceOf(owner.address)).to.be.equal(amountIn)

  // Get token price
  await PriceOracle.updateAssetPrice(XVS_TOKEN)
  const xvsPrice = await PriceOracle.getPrice(XVS_TOKEN)
  await PriceOracle.updateAssetPrice(WBNB_ADDR)
  const wbnbPrice = await PriceOracle.getPrice(WBNB_ADDR)

```



```

    await xvs.approve(XVSVaultConverter.address, amountIn)
    await XVSVaultConverter.convertExactTokensSupportingFeeOnTransferTokens(
        amountIn,
        ethers.utils.parseEther('0'),
        XVS_TOKEN,
        WBNB_ADDR,
        owner.address
    )

    const amountOut =
xvsPrice.mul(MANTISSA_ONE.add(incentive)).div(wbnbPrice).mul(amountIn).div(EXP_SCALE)
    expect(await xvs.balanceOf(XVSVaultTreasury.address)).to.be.equal(amountIn)
    expect(await wbnb.balanceOf(owner.address)).to.be.equal(amountOut)
});

it("convertForExactTokensSupportingFeeOnTransferTokens test", async function() {
    const { owner, signerXVS, XVSVaultTreasury, XVSVaultConverter, PriceOracle,
ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

    // set conversionConfig
    await ACM.giveCallPermission(XVSVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
    const incentive = ethers.utils.parseUnits('1', '18')
    const conversionConfig = [
        XVS_TOKEN,
        WBNB_ADDR,
        incentive,
        true
    ]
    await XVSVaultConverter.setConversionConfig(conversionConfig)

    // Preparation
    const amountOut = ethers.utils.parseEther('1')
    await wbnb.transfer(XVSVaultConverter.address, amountOut)
    await xvs.connect(signerXVS).transfer(owner.address,
ethers.utils.parseEther('1000'))

    expect(await wbnb.balanceOf(XVSVaultConverter.address)).to.be.equal(amountOut)
    expect(await
xvs.balanceOf(owner.address)).to.be.equal(ethers.utils.parseEther('1000'))

    // Get token price
    await PriceOracle.updateAssetPrice(XVS_TOKEN)
    const xvsPrice = await PriceOracle.getPrice(XVS_TOKEN)
    await PriceOracle.updateAssetPrice(WBNB_ADDR)
    const wbnbPrice = await PriceOracle.getPrice(WBNB_ADDR)

    const wbnbBefore = await wbnb.balanceOf(owner.address)

    await xvs.approve(XVSVaultConverter.address, ethers.utils.parseEther('1000'))
    await XVSVaultConverter.convertForExactTokensSupportingFeeOnTransferTokens(
        ethers.utils.parseEther('1000'),

```

```

        amountOut,
        XVS_TOKEN,
        WBNB_ADDR,
        owner.address
    )

    const amountIn =
amountOut.mul(EXP_SCALE).div(xvsPrice.mul(MANTISSA_ONE.add(incentive)).div(wbnbPrice))
    const wbnbAfter = await wbnb.balanceOf(owner.address)

    expect(await xvs.balanceOf(XVSVaultTreasury.address)).to.be.equal(amountIn)
    expect(wbnbAfter.sub(wbnbBefore)).to.be.equal(amountOut)
});

it("convert without conversionConfig init should failed", async function() {
    const { owner, signerXVS, XVSVaultTreasury, XVSVaultConverter, PriceOracle,
ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

    await expect(XVSVaultConverter.convertExactTokens(
        ethers.utils.parseEther('1'),
        ethers.utils.parseEther('0'),
        XVS_TOKEN,
        WBNB_ADDR,
        owner.address
    )).to.be.revertedWithCustomError(
        XVSVaultConverter, "ConversionConfigNotEnabled"
    )
});

it("convert zero amount should failed", async function() {
    const { owner, signerXVS, XVSVaultTreasury, XVSVaultConverter, PriceOracle,
ACM, wbnb, xvs } = await loadFixture(deployAndBindFixture)

    // set conversionConfig
    await ACM.giveCallPermission(XVSVaultConverter.address,
"setConversionConfig(ConversionConfig)", owner.address)
    const incentive = ethers.utils.parseUnits('1', '18')
    const conversionConfig = [
        XVS_TOKEN,
        WBNB_ADDR,
        incentive,
        true
    ]
    await XVSVaultConverter.setConversionConfig(conversionConfig)

    await expect(XVSVaultConverter.convertExactTokens(
        ethers.utils.parseEther('0'),
        ethers.utils.parseEther('0'),
        XVS_TOKEN,
        WBNB_ADDR,
        owner.address
    )).to.be.revertedWithCustomError(
        XVSVaultConverter, "InsufficientInputAmount"
    )
});

```

```

    )
  });
});
});

```

## 2. HardhatTestOutputXVS.md

```

XVSTokenConverter Unit Test
  XVSTokenConverter init unit test
    ✓ Initial state should equal with the params of constructor (2708ms)
  ACM and OnlyOwner unit test
    ✓ OnlyOwner unit test (48ms)
    ✓ ACM unit test (38ms)
  ConversionConfig unit test
    ✓ Set Zero Address should failed
    ✓ Set INCENTIVE too high should failed
  pauseConversion unit test
    ✓ Cannot convert tokens when paused
  sweepToken unit test
    ✓ sweepToken Zero Address should failed
  XVSVaultTreasury unit test
    ✓ fundXVSVault test (40ms)
  convertTokens unit test
    ✓ convertExactTokens test (122ms)
    ✓ convertForExactTokens test (103ms)
    ✓ convertExactTokensSupportingFeeOnTransferTokens test (99ms)
    ✓ convertForExactTokensSupportingFeeOnTransferTokens test (100ms)
    ✓ convert without conversionConfig init should failed
    ✓ convert zero amount should failed

```

14 passing (4s)

## 3. RiskFundConverter.t.js

```

const {time, loadFixture, impersonateAccount} = require("@nomicfoundation/hardhat-network-
helpers");
const { expect } = require("chai");
const { ethers } = require("hardhat");

describe("TokenConverter Unit Test", function () {
  const ZERO_ADDR = ethers.constants.AddressZero
  const WBNB_ADDR = "0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c"
  const VBNB_ADDR = "0xA07c5b74C9B40447a954e1466938b865b6BBEa36"

  const PRICE_ORACLE = "0x6592b5DE802159F3E74B2486b091D11a8256ab8A"

```

```

const CORE_POOL_COMPTRROLLER = "0xf6C14D4DFE45C132822Ce28c646753C54994E59C"
const POOL_REGISTRY = "0x9F7b01A536aFA00EF10310A162877fd792cd0666"

const Pool_Stablecoin_Comptroller = "0x94c1495cd4c557f1560Cbd68EAB0d197e6291571"
const Pool_Stablecoin_vTokens = [
  "0xCa2D81AA7C09A1a025De797600A7081146dceEd9",
  "0xc3a45ad8812189cAb659ad99E64B1376f6acd035",
  "0x5e3072305F9caE1c7A82F6Fe9E38811c74922c3B"
]
const Pool_Stablecoin_Assets = [
  "0x0782b6d8c4551B9760e74c0545a9bcd90bdc41E5",
  "0xd17479997F34dd9156Deef8F95A52D81d265be9c",
  "0x55d398326f99059fF775485246999027B3197955"
]

async function deployAndBindFixture() {
  const [owner, Alice, Bob, ...users] = await ethers.getSigners()

  await impersonateAccount(ZERO_ADDR);
  const signerZero = await ethers.getSigner(ZERO_ADDR)

  const RiskFundV2Factory = await ethers.getContractFactory('RiskFundV2', owner)
  const RiskFundV2 = await RiskFundV2Factory.deploy()

  const ACMFactory = await ethers.getContractFactory('MockACM', owner)
  const ACM = await ACMFactory.deploy()

  const RiskFundConverterFactory = await
ethers.getContractFactory('RiskFundConverter', owner)
  const RiskFundConverter = await
RiskFundConverterFactory.deploy(CORE_POOL_COMPTRROLLER)

  await RiskFundConverter.initialize(ACM.address, PRICE_ORACLE, RiskFundV2.address)
  await RiskFundConverter.setPoolRegistry(POOL_REGISTRY)

  await ACM.giveCallPermission(RiskFundConverter.address,
"setPoolsAssetsDirectTransfer(address[], address[][])", owner.address)
  await RiskFundConverter.setPoolsAssetsDirectTransfer([Pool_Stablecoin_Comptroller],
[Pool_Stablecoin_Assets])

  await RiskFundV2.connect(signerZero).transferOwnership(owner.address)
  await RiskFundV2.acceptOwnership()
  await RiskFundV2.setConvertibleBaseAsset(WBNB_ADDR)
  await RiskFundV2.setRiskFundConverter(RiskFundConverter.address)

  // return
  return { owner, Alice, Bob, signerZero, RiskFundV2, RiskFundConverter, ACM };
}

describe("RiskFundConverter init unit test", function() {
  it("Initial state should equal with the params of constructor", async function() {
    const { RiskFundV2, RiskFundConverter } = await
loadFixture(deployAndBindFixture)

```

```

        expect(await RiskFundConverter.priceOracle()).to.be.equal(PRICE_ORACLE)
        expect(await
RiskFundConverter.destinationAddress()).to.be.equal(RiskFundV2.address)
        expect(await RiskFundConverter.conversionPaused()).to.be.equal(false)
    });
});

describe("ACM and OnlyOwner unit test", function() {
    it("OnlyOwner unit test", async function() {
        const { Alice, RiskFundV2, RiskFundConverter } = await
loadFixture(deployAndBindFixture)
        await
expect(RiskFundConverter.connect(Alice).setPoolRegistry(ZERO_ADDR)).to.be.revertedWith("Own
able: caller is not the owner")
        await
expect(RiskFundV2.connect(Alice).setConvertibleBaseAsset(PRICE_ORACLE)).to.be.revertedWith(
"Ownable: caller is not the owner")
        await
expect(RiskFundV2.connect(Alice).setRiskFundConverter(RiskFundConverter.address)).to.be.rev
ertedWith("Ownable: caller is not the owner")
        await
expect(RiskFundV2.connect(Alice).setShortfallContractAddress(ZERO_ADDR)).to.be.revertedWith
("Ownable: caller is not the owner")
        await expect(RiskFundV2.connect(Alice).sweepToken(CORE_POOL_COMPROLLER,
WBNB_ADDR, 10000)).to.be.revertedWith("Ownable: caller is not the owner")
    });

    it("ACM unit test", async function() {
        const { Alice, RiskFundV2, RiskFundConverter, ACM } = await
loadFixture(deployAndBindFixture)
        const functionSig = "setPoolsAssetsDirectTransfer(address[], address[][])"
        await expect(RiskFundConverter.connect(Alice).setPoolsAssetsDirectTransfer([],
[[]])).to.be.revertedWithCustomError(
            RiskFundConverter, "Unauthorized"
        ).withArgs(Alice.address, RiskFundConverter.address, functionSig)
    });
});

describe("sweepToken unit test", function() {
    it("sweepToken Zero Address should failed", async function() {
        const { owner, RiskFundV2, ACM } = await loadFixture(deployAndBindFixture)
        await expect(RiskFundV2.sweepToken(ZERO_ADDR, ZERO_ADDR,
0)).to.be.revertedWithCustomError(
            RiskFundV2, "ZeroAddressNotAllowed"
        )
    });
});

describe("user interface unit test", function() {
    it("getPoolAssetReserve test", async function() {

```

```

        const { owner, RiskFundConverter, ACM } = await
loadFixture(deployAndBindFixture)
        const assetReserve = await
RiskFundConverter.getPoolAssetReserve(Pool_Stablecoin_Comptroller,
Pool_Stablecoin_Assets[0])
        expect(assetReserve).to.be.equal(0)
    });

    it("balanceOf test", async function() {
        const { owner, RiskFundConverter, ACM } = await
loadFixture(deployAndBindFixture)
        const balance = await RiskFundConverter.balanceOf(Pool_Stablecoin_Assets[0])
        expect(balance).to.be.equal(0)
    });

    it("updateAssetsState test", async function() {
        const { owner, RiskFundConverter, ACM } = await
loadFixture(deployAndBindFixture)
        const assetReserveBefore = await
RiskFundConverter.getPoolAssetReserve(Pool_Stablecoin_Comptroller,
Pool_Stablecoin_Assets[0])
        await RiskFundConverter.updateAssetsState(Pool_Stablecoin_Comptroller,
Pool_Stablecoin_Assets[0])
        const assetReserveAfter = await
RiskFundConverter.getPoolAssetReserve(Pool_Stablecoin_Comptroller,
Pool_Stablecoin_Assets[0])
        expect(assetReserveAfter.sub(assetReserveBefore)).to.be.equal(0)
    });

    it("postSweepToken test", async function() {
        const { owner, RiskFundConverter, ACM } = await
loadFixture(deployAndBindFixture)
        await RiskFundConverter.postSweepToken(Pool_Stablecoin_Assets[0], 0)
    });

    it("set Shortfall Zero should failed", async function() {
        const { owner, RiskFundV2, ACM } = await loadFixture(deployAndBindFixture)
        await
expect(RiskFundV2.setShortfallContractAddress(ZERO_ADDR)).to.be.revertedWithCustomError(
RiskFundV2, "ZeroAddressNotAllowed"
    )
    });
});
});

```

## 4. HardhatTestOutputRiskFund.md

TokenConverter Unit Test

RiskFundConverter init unit test

✓ Initial state should equal with the params of constructor (2341ms)

ACM and OnlyOwner unit test

✓ OnlyOwner unit test (52ms)

✓ ACM unit test

sweepToken unit test

✓ sweepToken Zero Address should failed

user interface unit test

✓ getPoolAssetReserve test

✓ balanceOf test

✓ updateAssetsState test

✓ postSweepToken test

✓ **set** shortfall Zero should failed

9 passing (3s)

## 11.2 External Functions Check Points

### 1. AbstractTokenConverter.sol\_output.md

**File: contracts/TokenConverter/AbstractTokenConverter.sol**

(Empty fields in the table represent things that are not required or relevant)

contract: AbstractTokenConverter is AccessControlledV8, IAbstractTokenConverter, ReentrancyGuardUpgradeable

Index	Function	Visibility	StateMutability	Permission Check	IsUserInterface	Unit Test	Notes
1	pauseConversion()	external		ACM		Passed	
2	resumeConversion()	external		ACM		Passed	
3	setPriceOracle(ResilientOracle)	external		onlyOwner		Passed	
4	setDestination(address)	external		onlyOwner		Passed	
5	setConversionConfig(ConversionConfig)	external		ACM		Passed	
6	convertExactTokens(uint256,uint256,address,address,address)	external			Yes	Passed	
7	convertForExactTokens(uint256,uint256,address,address,address)	external			Yes	Passed	
8	convertExactTokensSupportingFeeOnTransferTokens(uint256,uint256,address,address,address)	external			Yes	Passed	
9	convertForExactTokensSupportingFeeOnTransferTokens(uint256,uint256,address,address,address)	external			Yes	Passed	
10	sweepToken(address,address,uint256)	external		onlyOwner		Passed	
11	balanceOf(address)	public				Passed	
12	postSweepToken(address,uint256)	public				Passed	
13	getAmountOut(uint256,address,address)	public				Passed	
14	getAmountIn(uint256,address,address)	public				Passed	

## 2. XVSVaultTreasury.sol\_output.md

### File: contracts/ProtocolReserve/XSVVaultTreasury.sol

(Empty fields in the table represent things that are not required or relevant)

contract: XSVVaultTreasury is AccessControlledV8

Index	Function	Visibility	StateMutability	Permission Check	IsUserInterface	Unit Test	Notes
1	setXSVVault(address)	external		onlyOwner		Passed	
2	fundXSVVault(uint256)	external		ACM		Passed	
3	initialize(address,address)	public				Passed	

## 3. XSVVaultConverter.sol\_output.md

### File: contracts/TokenConverter/XSVVaultConverter.sol

(Empty fields in the table represent things that are not required or relevant)

contract: XSVVaultConverter is AbstractTokenConverter

Index	Function	Visibility	StateMutability	Permission Check	IsUserInterface	Unit Test	Notes
1	initialize(address,ResilientOracle,address)	public				Passed	
2	updateAssetsState(address,address)	public			Yes	Passed	
3	balanceOf(address)	public	view		Yes	Passed	

## 4. RiskFundV2.sol\_output.md

### File: contracts/ProtocolReserve/RiskFundV2.sol

(Empty fields in the table represent things that are not required or relevant)

contract: RiskFundV2 is Ownable2StepUpgradeable, AccessControlledV8, RiskFundV2Storage, IRiskFund, ReentrancyGuardUpgradeable

Index	Function	Visibility	StateMutability	Permission Check	IsUserInterface	Unit Test	Notes
1	setConvertibleBaseAsset(address)	external		onlyOwner		Passed	
2	setRiskFundConverter(address)	external		onlyOwner		Passed	
3	setShortfallContractAddress(address)	external		onlyOwner		Passed	
4	transferReserveForAuction(address,address,uint256)	external		onlyShortFall			
5	sweepToken(address,address,uint256)	external		onlyOwner		Passed	
6	updatePoolState(address,address,uint256)	public		onlyRiskFundConverter		Passed	



## 5. RiskFundConverter.sol\_output.md

### File: contracts/TokenConverter/RiskFundConverter.sol

(Empty fields in the table represent things that are not required or relevant)

contract: RiskFundConverter is AbstractTokenConverter

Index	Function	Visibility	StateMutability	Permission Check	IsUserInterface	Unit Test	Notes
1	setPoolRegistry(address)	external		onlyOwner		Passed	
2	setPoolsAssetsDirectTransfer(address[],address[])	external		ACM		Passed	
3	getPoolAssetReserve(address,address)	external	view		Yes	Passed	
4	initialize(address,ResilientOracle,address)	public				Passed	
5	updateAssetsState(address,address)	public			Yes	Passed	
6	postSweepToken(address,uint256)	public			Yes	Passed	
7	balanceOf(address)	public	view		Yes	Passed	



<https://medium.com/@FairyproofT>



<https://twitter.com/FairyproofT>



<https://www.linkedin.com/company/fairyproof-tech>



[https://t.me/Fairyproof\\_tech](https://t.me/Fairyproof_tech)



Reddit: <https://www.reddit.com/user/FairyproofTech>

