Ultrade CODEX

Ultrade Solana & Algorand Cross Chain Testing

Smart Contract Audit Report

DATE: 12th December 2024 | **PREPARED BY:** Entersoft Pty Ltd

Contents

Revision History & Version Control **3**
1. Executive Summary **3**
2. Overview **4**
2.1 Project Overview 4
2.2. Scope 4
2.3 Summary of Findings 4
2.4 Security Level References 4
3. Comprehensive Analysis findings:Manual Testing **5**
4. Tested For Scenarios **6**
4.1 Deposit Solana to Codex Cross chain Duplication 6
4.2 Withdraw Codex to Solana Cross chain Duplication 10
4.3 Manipulating Chain Id in VAA during Deposit 15
4.4 Manipulating Amount in VAA during Deposit 21
4.5 Manipulating Address in VAA during Deposit 26
4.6 Manipulating “Deposit” String in VAA during Deposit 35
4.7 Manipulating Token Address in VAA during Withdraw 41
4.8 Manipulating “Withdraw” String in VAA during Withdraw 47
4.9 Manipulating Amount in VAA during Withdraw 53
4.10 Manipulating ChainId in VAA during Withdraw 60
5. Auditing Approach and Methodologies Applied **68**
6. Limitations on Disclosure and Use of this Report **69**

Revision History & Version Control

Version	Date	Author(s)	Description
3.0	12-December-2024	Gurkirat singh, Tharun Betina	Final Report

1. Executive Summary

Entersoft has conducted a cross chain smart contract testing of the Ultrade Solana Algorand project through a comprehensive smart contract audit. The primary objective was to identify potential vulnerabilities and risks within the codebase, ensuring adherence to industry-leading standards while prioritizing security, reliability, and performance. Our focus was on prompt and efficient identification and resolution of vulnerabilities to enhance the overall robustness of the smart contract.

Testing Methodology:

Our testing methodology adheres to industry standards and best practices, integrating partially implemented OWASP and NIST SP 800 standards for encryption and signatures.

We have performed a detailed manual analysis, adherence to industry standards, and the use of a comprehensive toolset. Our approach ensured a thorough evaluation within the designated code files.

Findings and Security Posture:

Our audit process emphasized a comprehensive and meticulous approach to security analysis, prioritizing depth and precision throughout the evaluation. The audit was conducted from December 2, 2024, to December 12, 2024. During this timeframe, Entersoft thoroughly reassessed and validated the security posture of the Cross-Chain functionality for Solana and Algorand. Following this detailed evaluation, the system was designated as 'Secure,' signifying the absence of identified vulnerabilities and the robustness of the codebase in mitigating potential security threats.

2. Overview

2.1 Project Overview

Entersoft has meticulously tested cross chain transactions which involved Solana, Algorand and Wormhole from December 2, 2024, and December 12, 2024 with a primary focus on code files integral to blockchain functionality, emphasizing vulnerabilities in associated gas claiming. The working of basic functionalities was also tested during the review.

2.2. Scope

The audit scope covers the CODEX-dev smart contract available in the GitHub private repository:

Github	https://github.com/ultrade-org/CODEX
Branch	Dev Branch
Commit ID	e623676a0a6e8d69337019137d0f6dc1b8e4d1a2

OUT-OF-SCOPE: External contracts, other imported smart contracts.

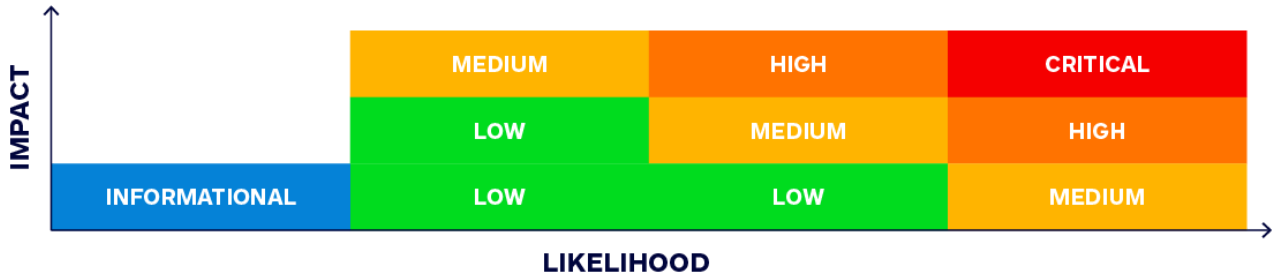
2.3 Summary of Findings

The following table is the summary of findings, which summarizes the overall risks identified during the audit.

Critical	High	Medium	Low	Informational
0	0	0	0	0

2.4 Security Level References

Every vulnerability in this report was assigned a severity level from the following classification table:



3. Comprehensive Analysis findings:Manual Testing

Phase 1: Dynamic Analysis

Dynamic analysis was conducted, which involved a more hands-on approach to scrutinizing the intended functionality and security of the cross chain transactions. Further, we devised a comprehensive suite of manual tests and manipulated raw VAA to validate the expected behavior of the smart contracts under various scenarios.

Processes and Test Cases:

Our dynamic analysis systematically explored the cross-chain functionalities of the protocol, concentrating on critical areas susceptible to vulnerabilities. During testing, when a deposit transaction occurred, the transaction hash was retrieved from Testnet Devnet, and the raw VAA was obtained from Wormhole Scan. The VAA was decrypted to reveal its payload, which was then altered and processed on Algorand to observe whether the transaction succeeded or reverted. Similarly, for withdrawal transactions from Algorand to the Solana chain, the process was reversed to ensure comprehensive validation.

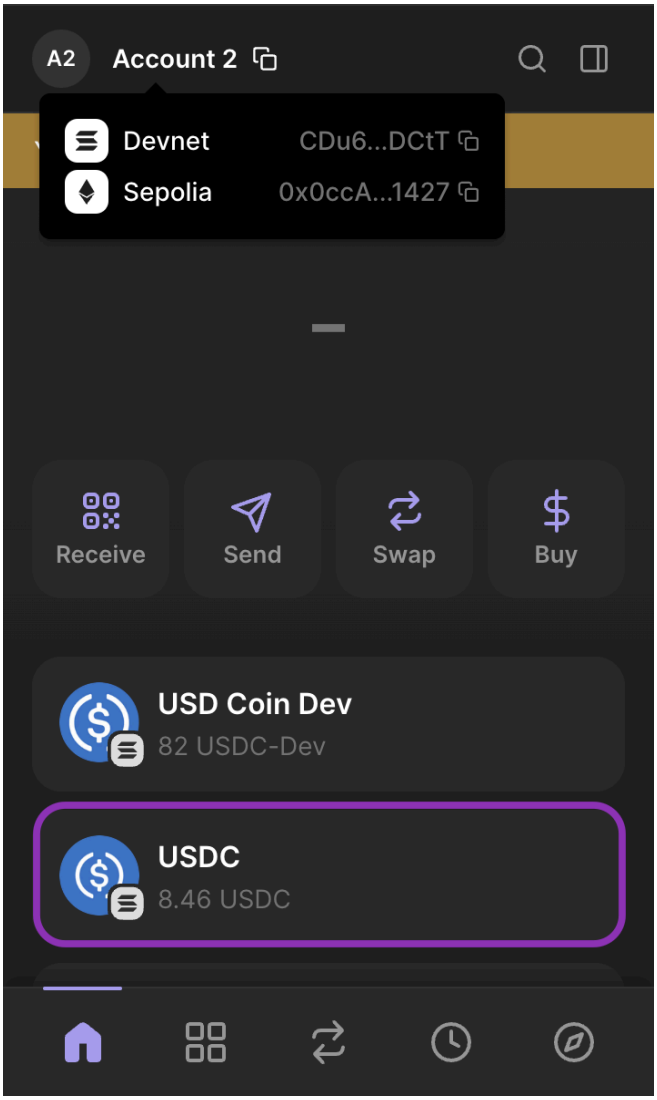
Throughout the dynamic analysis phases, our team dedicated substantial efforts to meticulously review the codebase, identify vulnerabilities, and develop robust manual test cases to assess the security posture of the protocol comprehensively.

4. Tested For Scenarios

4.1 Deposit Solana to Codex Cross chain Duplication

Result	Pass
Description	Testing Cross-Chain Deposit Using Wormhole VAA In this scenario, we are validating the behavior of cross-chain deposit transactions facilitated by the Wormhole protocol. The focus is on the manual verification of the VAA to ensure the prevention of duplicate transactions.
Function	Deposit
Steps Performed	1. **Check Balance of the User:** ● Verify initial token balances of the user in EOA and the codex contract. 2. **Deposit Function Execution:** ● Execute the deposit function on the Solana chain. ● Capture the transaction hash for further analysis. 3. **Collection Correct Raw VAA And Executing it on Solana:** ● Retrieve the original VAA from Wormhole scan for the deposit transaction. ● Submit the original, unaltered VAA to verify the deposit proceeds as intended. ● Check if the user's EOA balance is updated correctly. 4. **Duplicate Deposit Attempt:** ● The same deposit transaction ID and VAA are reused to attempt a duplicate deposit on the Algorand network. ● This results in an error, confirming that the system prevents replay attacks or duplicate transaction executions. 5. **Final Validation:** ● Ensure the raw VAA passes but the duplicated vaa fails and does not alter balances. ● Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex**

User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance



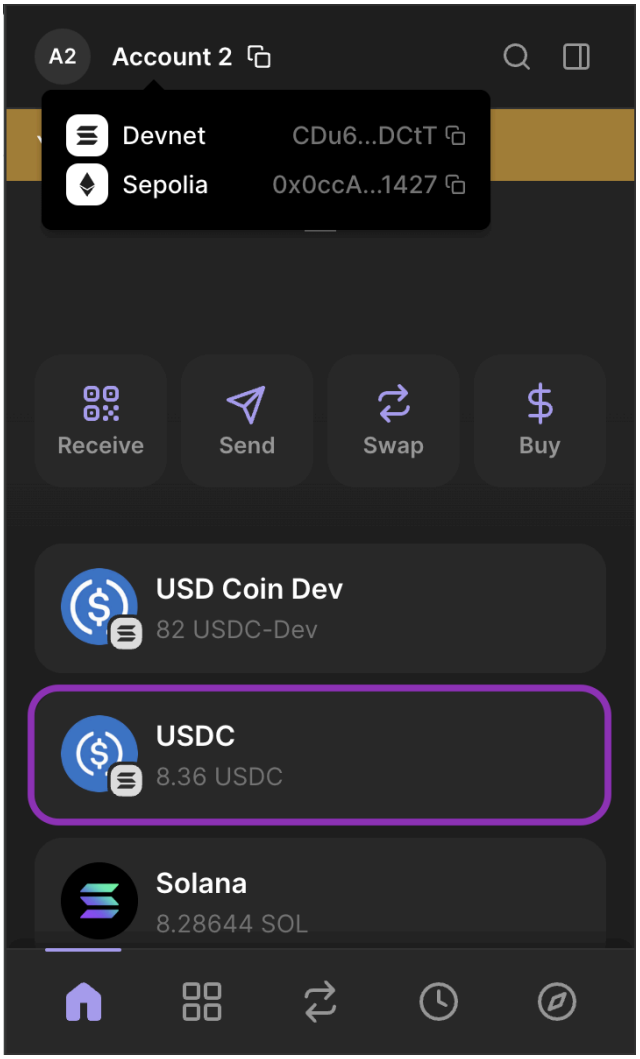
Balance in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 43435450555344430000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
' 32
userBalance 500000
```

Step 2: Deposit Function Execution

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run depositUSDC

> depositUSDC
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --depositUSDC

secp256k1 unavailable, reverting to browser version
depositUSDC called
token account PublicKey [PublicKey(22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87)] {
  _bn: <BN: f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc>
}


  0 passing (0ms)

_config PublicKey [PublicKey(3pp7kNQMeGcFQ7BNfF5uWR9pU2rgDjgK7KXGoA5gfwHQ)] {
  _bn: <BN: 29f6f24d8148cb1b0e60424c1a8834046d7a572e82e3713e4d01e57f14af4d6f>
}
vaultKey PublicKey [PublicKey(3c4YrWQEqujUEfSdoHyji5te7rtXLWj6Xmi7BUXbyXyi)] {
  _bn: <BN: 26b2b71b78a90614412b39fdc371d659abee46541bad88ad30426502bf4ec671>
}
payload <Buffer 64 65 70 6f 73 69 74 a6 be 70 40 7b 6f 5e 03 3e 67 75 ef f7 d2 53 14 cd e5 7c 0d b8 29 2f 5d 54 69 1a 2a 68 93
 dc c4 43 43 54 50 55 53 44 43 00 00 00 ... 117 more bytes>
```

USDC Deducted from the wallet



Txn ID — NvVKtfrkfY71uNK6xNaA9wrZ9c9kaiZbmEk77g6JQ3tiwkM6BRfEaR1xentuLsfkLTHXuXKTv4LEGUSnoVqVKWN

Overview — Inspect — Refresh

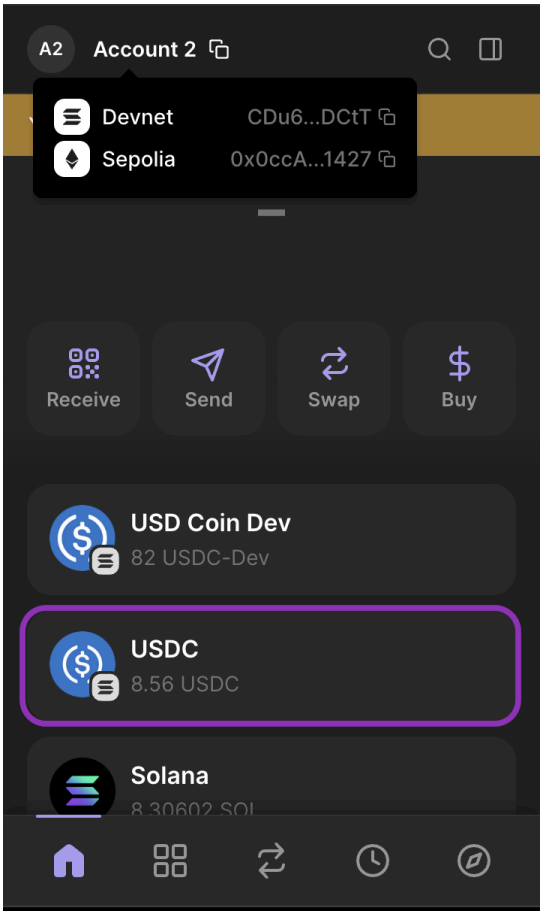
Signature	NvVKtfrkfY71uNK6xNaA9wrZ9c9kaiZbmEk77g6JQ3tiwkM6BRfEaR1xentuLsfkLTHXuXKTv4LEGUSnoVqVKWN
Result	Success
Timestamp	Dec 11, 2024 at 21:52:32 India Standard Time
Confirmation Status	FINALIZED
Confirmations	MAX
Slot	346,264,599
Recent Blockhash	9TauMf89aNY84mWnhnLzTSb3V7rTf1FzKijaPHxyTtGY
Fee (SOL)	◎0.000005
Compute units consumed	94,437
Transaction Version	LEGACY

Step 3: Collected Correct Raw VAA And Executing it on Algorand

Entering the correct vaa:-

AQAAAAABAJs0lsJQ3prHJDK5/6ue1syW99cyTlo/oVNwXCSIXoE4YBEblKDvexRuTpnD8NED7eRpVhWsN7c8JLTUfiRKY2wAZ1m8SAAAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAAABgBZGVwb3NpdKa+cEB7b14DPmd17/fSUxTN5XwNuCkvXVRpGipok9zEQ0NUUFVTREMAAAAAAAAA

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXWLRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC1AADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC1AADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y
sig_addr 86525623 Z7AISPXE4YFE5MU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c638000000000
```

	• Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex** User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance  Balance in codex (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 1 login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT assest Chain id 65537 assest address 4343545055534443000000000000000000000000000000000000000000000000 b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t'\xc0" 32 userBalance 600000 **Step 2: Withdrawal Function Execution** (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 727096157 before algo amount of codex 2000000 codex min balance 409100 before algo amount of msg processor 2000000 1 CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT dest_private_key TtNrm5b1TvgbW5HhkyifVLeRoifvk6LX64c3V4ffv19Zky7F3gWJ7FxEoUkM6JCkkJ5rp7DWPXDXZAQTCaY6aEP 4343545055534443000000000000000000000000000000000000000000000000 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 1 emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL KVNGXKQ4OCKMU5Y login CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1 super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 1 236 1 b'TX{"Login_Address": "CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT", "Login_ChainId": "1", "Token_Address": "4343545055534443 0000000000000000000000000000000000000000000000000", "Receiver_Address": "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87", "Receiv er_ChainId": "1", "Amount": "50000", "Withdraw_Fee": "False"}pr5wQHtvXgM+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQAAAAAAAAAAUNDVFBVU0RDA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADzxpeFz1EYlzZf4LHDc//FG/NQdtAg7wL3NHMWYIAvwAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMN QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' userBalance 550000 after algo amount of codex 2000000 after algo amount of msg processor 2000000 Balance after withdrawal

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08dp'
32
userBalance 550000
```

Txn Id:- VDHF66XDSM4NTHT7MVX3OJUC3PBLL6IX23CTLXGCHUQXK6NNC45A

Application ID	On Completion	Fee
727096157	NoOp	0.007

Sender AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU Copy

Transaction details

Group ID	Ks1Cga1uWf9atFfn0bHYLMP7I4NBfATaDd5myTiza/4=	Copy
Transaction ID	VDHF66XDSM4NTHT7MVX3OJUC3PBLL6IX23CTLXGCHUQXK6NNC45A	Copy
Block	46629880	
Timestamp	Dec. 11, 2024, 10:13 a.m. (0 minutes ago)	

Step 3: Collected Correct Raw VAA And Executing it on Solana

Entering the correct vaa:-

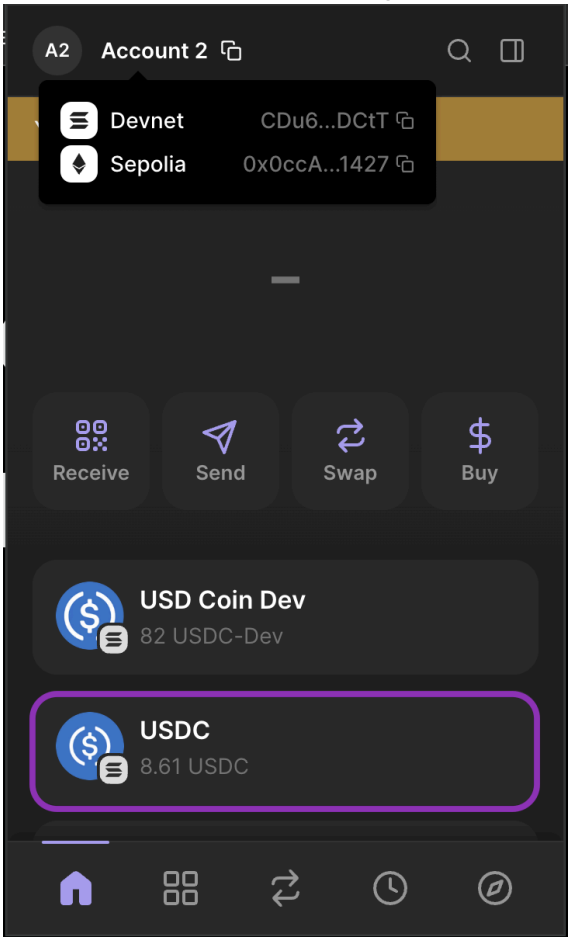
AQAAAAABAJxfAibzqJmVXmkChMSMEpJ1mAdiB6/OFzYdoxUY3Cv7YKSrLzamxsjNB2+dIW73HYrqOwG+7Yr6L+GtXUG0GZwBZ1llswAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACwAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAKjOX3rjkzjZnn9lb7cmgtvCtfkX1sU13MI9IXV5rRc60Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA

```
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAJxfAibzqJmVXmkChMSMEpJ1mAdiB6/OFzYdoxUY3Cv7YKSrLzamxsjNB2+dIW73HYrqOwG+7Yr6L
  +GtXUG0GZwBZ1llswAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACwAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAKjOX3rjkzj
  Znn9lb7cmgtvCtfkX1sU13MI9IXV5rRc60Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

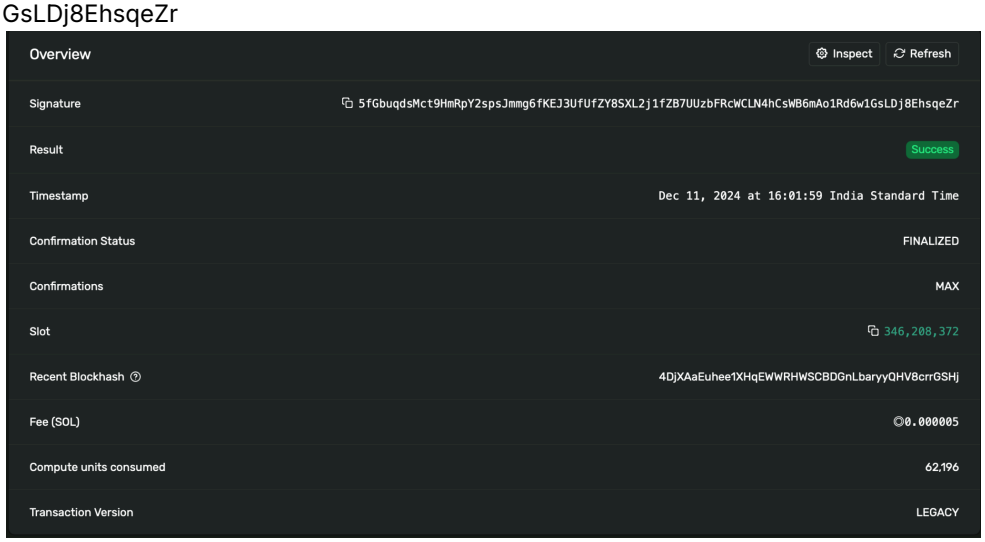
Executing Withdraw in solana with correct vaa

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (0ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 9c 5f 02 26 f3 a8 99 95 5e 69 02 84 c4 8c 12 92 75 98 07 62 07 af ce 17 36 1d a3 15 18
  dc 2b fb 60 a4 ab 2f 36 a6 c6 c8 cd 07 6f ... 259 more bytes>
  response [
    {
      signature: '63H8eGvTTy8Dt6UYkRZGBe86ZFkZAsmX8iTbBJksLNoYcwedDU2bNaZXLEwdjXXLvsXJvKEXNG9WuVLjd47JQxmo',
      response: { context: [Object], value: [Object] }
    },
    {
      signature: '5JPXW2B83JWTiFRiysekBhygFkJMkFueuzDf8qcthYCegXvo6VVRKw2iZkceVtFkzv5FHZJ8boSNwLA4SFU6gQVj',
      response: { context: [Object], value: [Object] }
    }
  ]
  createWithdrawTokenInstruction is called
  token PublicKey [PublicKey(4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU)] {
    _bn: <BN: 3b442cb3912157f13a933d0134282d032b5ffecd01a2dbf1b7790608df002ea7>
  }
  receivedAcc PublicKey [PublicKey(52MEs9viERbxSmyH91Wu1He2bQSiEJRzEPeQeYtAdgej)] {
    _bn: <BN: 3bc7367f290dc373421b0c50a997198f02c3af78b2c1c2b9df6346323b9cca58>
  }
  posted PublicKey [PublicKey(B8d3wQj2mqEzokyrAghyZ8bd1WbwK7MNQDthNw83Ltj7)] {
    _bn: <BN: 9688b2919f0142c960d8065b48860b8c71fcf8f96af969d2333c67fd910ea62e>
  }
```

Balance updated of the user EOA gets updates with correct vaa



Transaction id on Solana is:-
5fGbuqdsMct9HmRpY2spsJmmg6fKEJ3UfUfZY8SXL2j1fZB7UUzbFRcWCLN4hCsWB6mAo1Rd6w1GsLDj8EhsqeZr



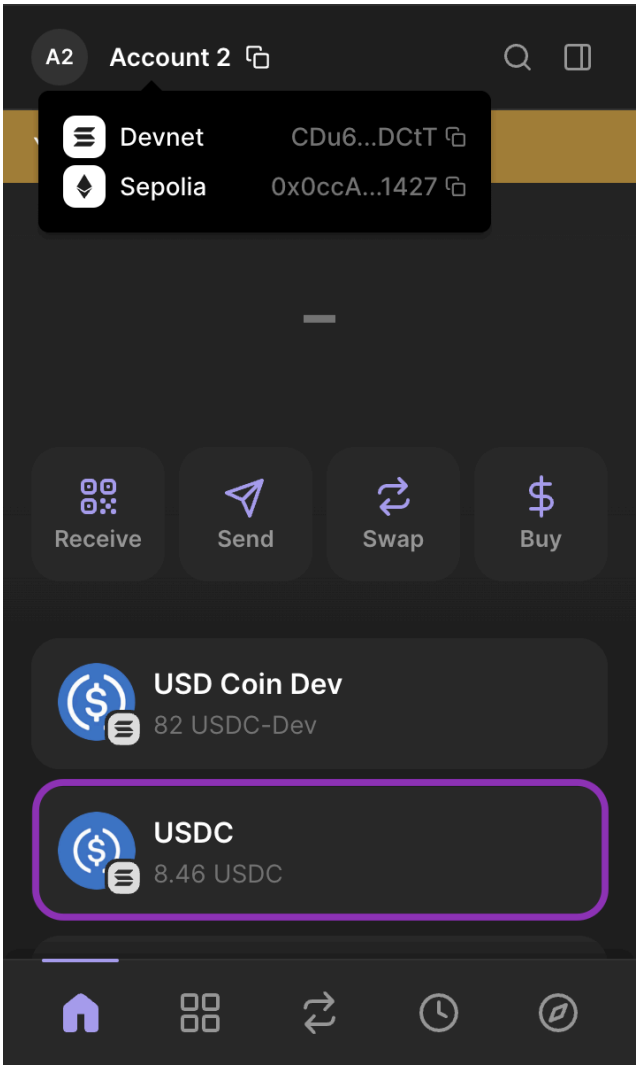
Overview	Inspect / Refresh
Signature	5fGbuqdsMct9HmRpY2spsJmmg6fKEJ3UfUfZY8SXL2j1fZB7UUzbFRcWCLN4hCsWB6mAo1Rd6w1GsLDj8EhsqeZr
Result	Success
Timestamp	Dec 11, 2024 at 16:01:59 India Standard Time
Confirmation Status	FINALIZED
Confirmations	MAX
Slot	346,208,372
Recent Blockhash	4DjXAaEuhee1XHqEWWRHWSCBDGnLbaryyQHV8crrGSHj
Fee (SOL)	◎0.000005
Compute units consumed	62,196
Transaction Version	LEGACY

Steps 4: Duplicate Withdrawal Attempt

Entering the same raw VAA again
Getting Below error

```
● gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (0ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 15 83 60 e9 50 9c 81 76 04 52 37 c2 50 c3 42 b4 22 c6 ce 85 51 08 da b2 df f4 1b 78 a2
  e5 ba f9 7d f0 ed 7f 95 fb 92 34 fa 8c a9 ... 259 more bytes>
  SendTransactionError: failed to send transaction: Transaction simulation failed: Error processing Instruction 0: custom progra
  m error: 0x0
      at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
  DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
      at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
  dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
    logs: [
      'Program 3u8hJUVTA4jH1wYAyUur7FFZVQ8H635K3tSHHF4ssjQ5 invoke [1]',
      'Program log: Instruction: LegacyPostVaa',
      'Program 11111111111111111111111111111111 invoke [2]',
      'Allocate: account Address { address: AzAkTMNYtwieL7RwYGL16a1ek9F82MZsxD4sj5BrRGEf, base: None } already in use',
      'Program 11111111111111111111111111111111 failed: custom program error: 0x0',
      'Program 3u8hJUVTA4jH1wYAyUur7FFZVQ8H635K3tSHHF4ssjQ5 consumed 11128 of 200000 compute units',
      'Program 3u8hJUVTA4jH1wYAyUur7FFZVQ8H635K3tSHHF4ssjQ5 failed: custom program error: 0x0'
    ]
  }
```

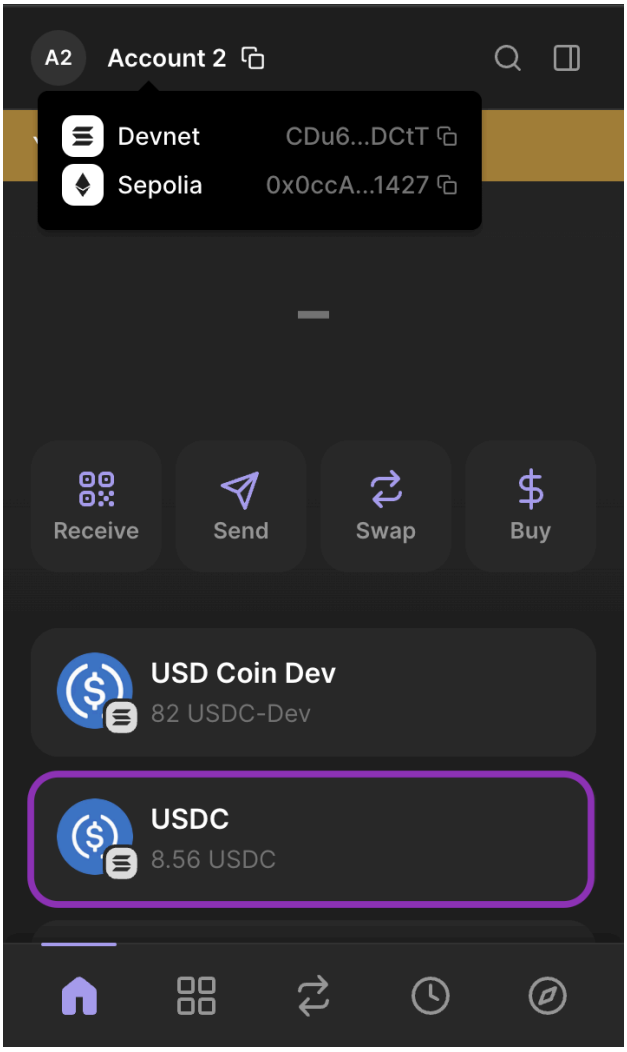
EOA Balance didn't get updated

4.3 Manipulating Chain Id in VAA during Deposit

Result	**Pass**
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect chainId) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Deposit
Steps Performed	1. **Check Balance of the User:** • Verify initial token balances of the user in EOA and the codex contract. 2. **Deposit Function Execution:** • Execute the deposit function on the Solana chain. • Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** • Retrieve the original VAA from Wormhole scan for the deposit transaction.

	● Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** ● Modify the Chain Id in the payload from the correct value 0000000000000000000000000000000000000000000000000000000000000001 to a wrong value 0000000000000000000000000000000000000000000000000000000000000002 . ● Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** ● Encrypt the manipulated VAA and submit it to the Algorand network manually for verification. ● Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** ● Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended. ● Check if the user's balance is updated correctly. 7. **Final Validation:** ● Ensure the manipulated VAA fails and does not alter balances. ● Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex** User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance



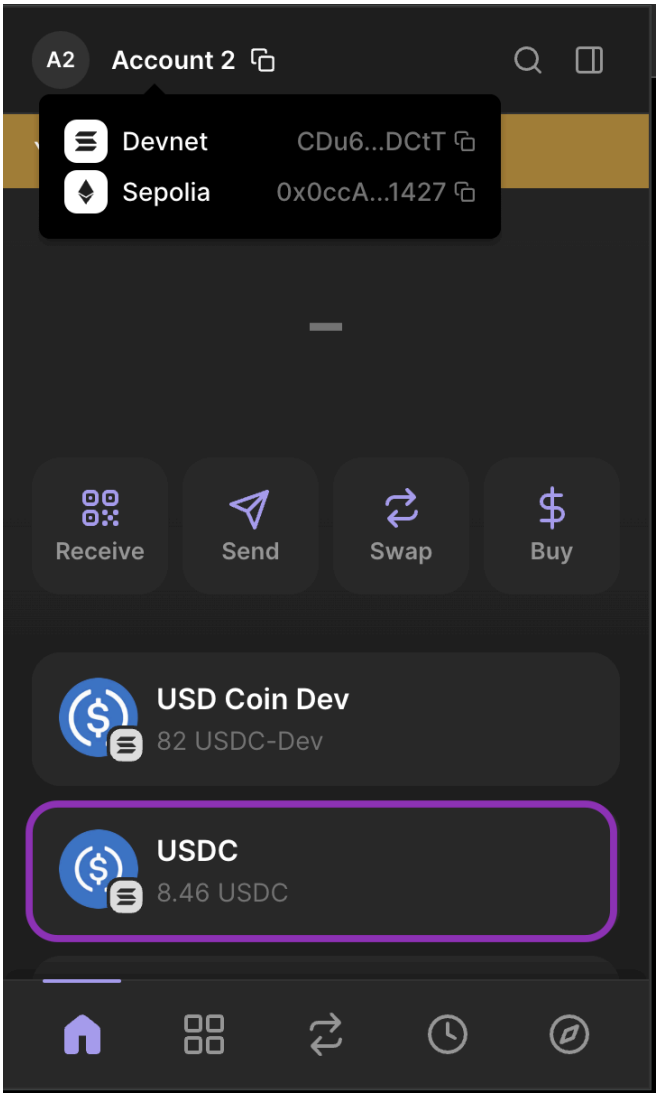
Balance in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t\xeb\x
10' 32
userBalance 650000
```

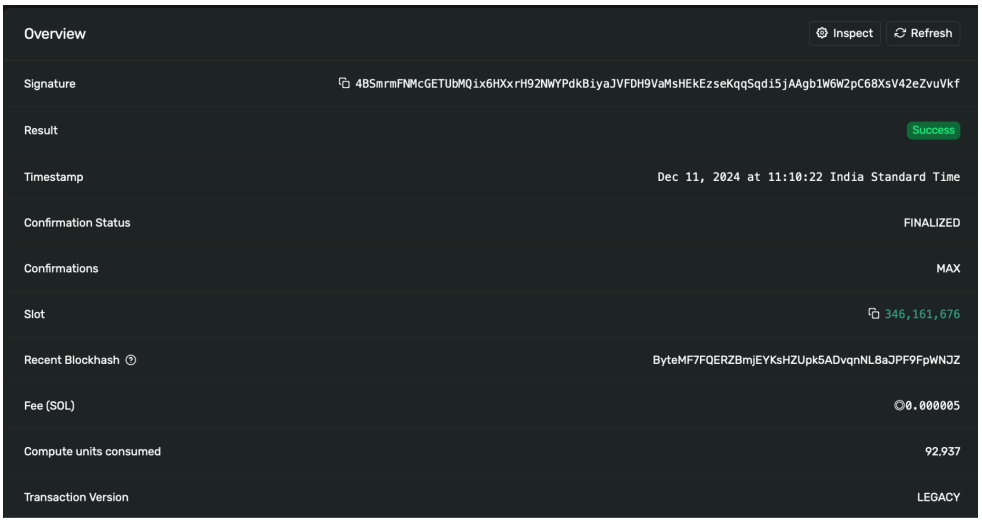
Step 2: Deposit Function Execution

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run depositUSDC

> depositUSDC
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --depositUSDC

secp256k1 unavailable, reverting to browser version
depositUSDC called
token account PublicKey [PublicKey(22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87)] {
  _bn: <BN: f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc>
}


  0 passing (0ms)

_config PublicKey [PublicKey(3pp7kNQMeGcFQ7BNfF5uWR9pU2rgDjgK7KXGoA5gfwHQ)] {
  _bn: <BN: 29f6f24d8148cb1b0e60424c1a8834046d7a572e82e3713e4d01e57f14af4d6f>
}
vaultKey PublicKey [PublicKey(3c4YrWQEqujUEfSdoHyji5te7rtXLWj6Xmi7BUXbyXyi)] {
  _bn: <BN: 26b2b71b78a90614412b39fdc371d659abee46541bad88ad30426502bf4ec671>
}
payload <Buffer 64 65 70 6f 73 69 74 a6 be 70 40 7b 6f 5e 03 3e 67 75 ef f7 d2 53 14 cd e5 7c 0d b8 29 2f 5d 54 69 1a 2a 68 93
 dc c4 43 43 54 50 55 53 44 43 00 00 00 ... 117 more bytes>
```

USDC Deducted from the wallet



Txn ID —
4BSmrmFNMcGETUbMQix6HXxrH92NWYPdkBiyaJVFDH9VaMsHEkEzseKqqSqdi5jAAgb1W6W2pC68XsV42eZvuVkf



Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa

AQAAAAABAD+jrfdZhdJDchXqnD7XUW3nFUJHbAxmwYMIzP/vy+McK5kKRS9WR/8R5BWPnadsWEFjgi9w+/mqqDlwb6NxDBgBZ1klxgAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAAABUBZGVwb3NpdKa+cEB7b14DPmd17/fSUxTN5XwNuCkvXVRpGipok9zEQ0NUUFVTREMAAAAAAAA

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=')
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000
```

	5WRNM3ZPD... 46624215 AFIA...64JU → 727096228 Application Call 1 minute CH6M...N2CE → 727096157 Application Call 1 minute

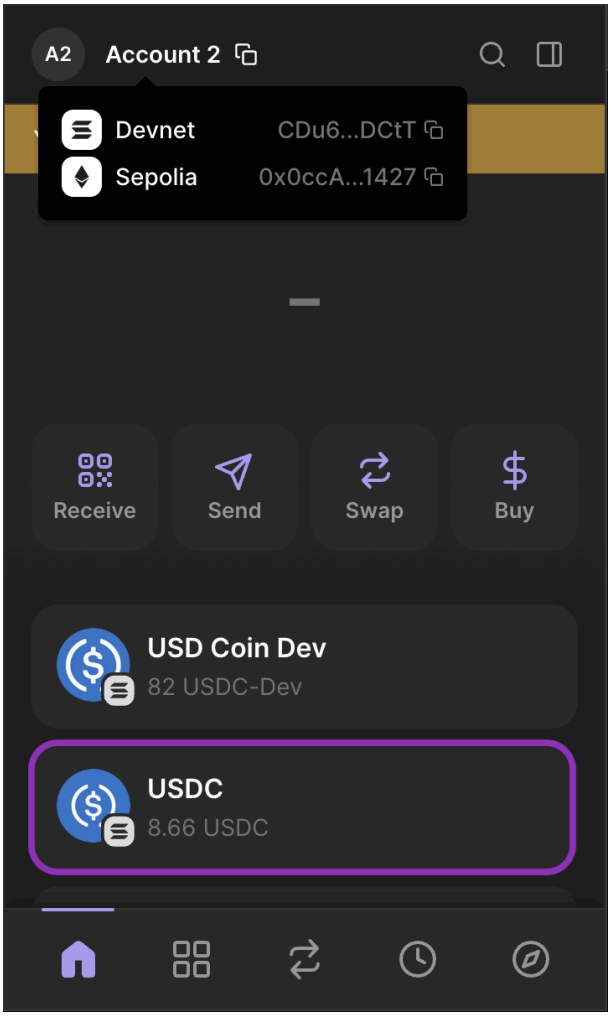
4.4 Manipulating Amount in VAA during Deposit

Result	Pass
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect amount) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Deposit
Steps Performed	1. **Check Balance of the User:** - Verify initial token balances of the user in EOA and the codex contract. 2. **Deposit Function Execution:** - Execute the deposit function on the Solana chain. - Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** - Retrieve the original VAA from Wormhole scan for the deposit transaction. - Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** - Modify the Amount in the payload from the correct value 10000 to a wrong value 50000 . - Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** - Encrypt the manipulated VAA and submit it to the Algorand network manually for verification. - Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** - Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended. - Check if the user's balance is updated correctly. 7. **Final Validation:** - Ensure the manipulated VAA fails and does not alter balances. - Confirm the correct VAA succeeds, incrementing the user's balance.

Observation

Step 1: Check the balance of the user in EOA and Codex

User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance



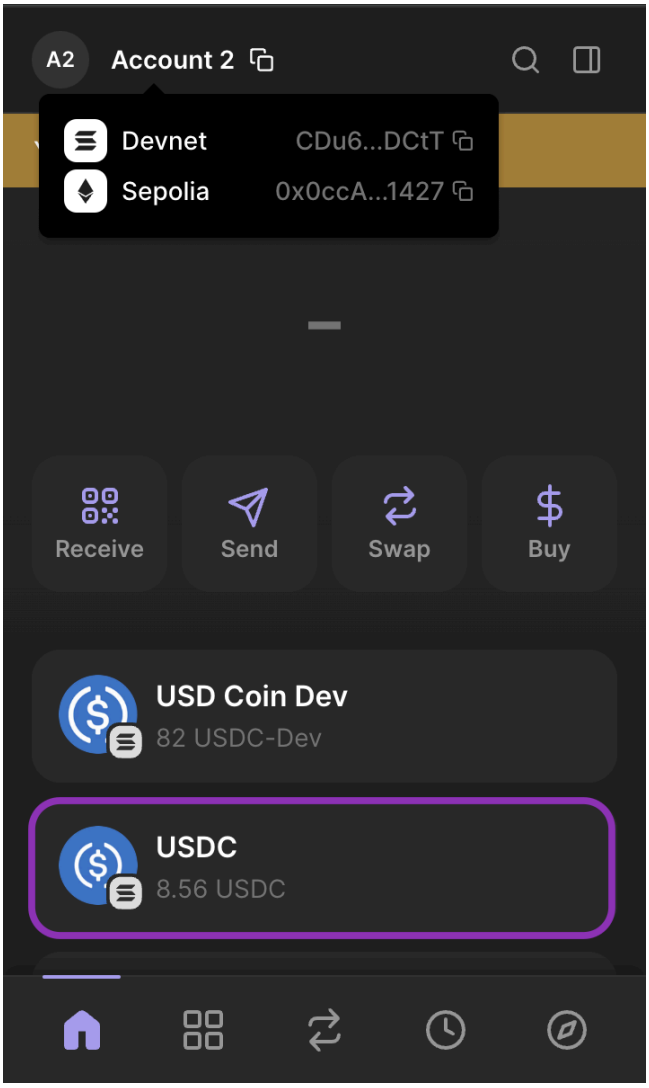
Balancel in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08dp'
32
userBalance 550000
```

Step 2: Deposit Function Execution

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run depositUSDC

> depositUSDC
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --depositUSDC

secp256k1 unavailable, reverting to browser version
depositUSDC called
token account PublicKey [PublicKey(22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87)] {
  _bn: <BN: f3c69785cf51189736 5fe0b1c373ffc51bf35076d020ef02f734731660802fc>
}


  0 passing (0ms)

_config PublicKey [PublicKey(3pp7kNQMeGcFQ7BNfF5uWR9pU2rgDjgK7KXGoA5gfwHQ)] {
  _bn: <BN: 29f6f24d8148cb1b0e60424c1a8834046d7a572e82e3713e4d01e57f14af4d6f>
}
vaultKey PublicKey [PublicKey(3c4YrWQEqujUEfSdoHyji5te7rtXLWj6Xmi7BUXbyXyi)] {
  _bn: <BN: 26b2b71b78a90614412b39fdc371d659abee46541bad88ad30426502bf4ec671>
}
payload <Buffer 64 65 70 6f 73 69 74 a6 be 70 40 7b 6f 5e 03 3e 67 75 ef f7 d2 53 14 cd e5 7c 0d b8 29 2f 5d 54 69 1a 2a 68 93
 dc c4 43 43 54 50 55 53 44 43 00 00 00 ... 117 more bytes>
```

USDC Deducted from the wallet



Txn ID — 5pWB46WFNzGmjbhdj14QJzjTWTknYfUKoUpMdJFRtHG8vHWPUf1qJvLLPV4La3oxcEotpQcen2PXkrfkpdkDn8T3

Overview | Inspect | Refresh

Signature	5pWB46WFNzGmjbhdj14QJzjTWTknYfUKoUpMdJFRtHG8vHWPUf1qJvLLPV4La3oxcEotpQcen2PXkrfkpdkDn8T3
Result	Success
Timestamp	Dec 11, 2024 at 10:31:57 India Standard Time
Confirmation Status	FINALIZED
Confirmations	MAX
Slot	346,155,525
Recent Blockhash	Gt6qh7osCrXQDfccqbSRZir7TUyR4yTTWKPB3pxtrAbj
Fee (SOL)	◎0.000005
Compute units consumed	95,937
Transaction Version	LEGACY

Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa
AQAAAAABAI8zLaPaX9A+L0lGo8Dsx9wxS/E+7ARt9IRLFf0ISzK1Ye83rg3ckOppNwNKEdEnBbfVx5HF2+2rTmX9nYUXZaYAZ1kcxQAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAABQBZGVwb3NpdKa+cEB7b14DPmd17/fSUxTN5XwNuCkvXVRpGipok9zEQ0NUUFVTREMAAAAAAA

```
if __name__ == "__main__":
    vaaString = "AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwhFf+3gBa07EoAOk4DyD47SCB0BZ1gZ/
    QAAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAAABIBd2l0aGRyYXemvnBAe29eAz5nde/
    30lMUzeV8DbgpL11UaRoqaJPcxENDVFBVU0RDAAAAAAAA
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBB
hJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VA
CVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4F
BCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000006775617264696 16e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x108926e40>, <algosdk.transaction.Applicat
ionCallTxn object at 0x108ee1880>] ['R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y', 'Z7AISPXE4YFESMU35JXBEM6JZSK
TVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1 434354505553444300000000000000000000000000000000000000000000000 65537 THIS THE
 DATA WE ARE PRINTING
txns [<algosdk.transaction.ApplicationCallTxn object at 0x108926e40>, <algosdk.transaction.ApplicationCallTxn object at 0x108e
e1880>, <algosdk.transaction.ApplicationCallTxn object at 0x10902efc0>]
result {'confirmed-round': 46623591, 'pool-error': '', 'txn': {'lsig': {'l': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC
iAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCD
UDNAQlCDUEQv+0Iok='}, 'txn': {'apaa': ['dmVyaWZ5U2lncw==', 'AI8zLaPaX9A+L0lGo8Dsx9wxS/E+7ARt9IRLFf0lSzK1Ye83rg3ckOppNwNKEdEnBb
fVx5HF2+2rTmX9nYUXZaYA', 'E5R71IsY5T/a7ud/NHM5Gscnxjg=', 'byyC3GdWycwV6UoD83afl3ZuCgpoy0IJCtDRJ/4mBGw='], 'apat': ['R7ENA46PT7
Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ'], 'apid': 8652
5623, 'fv': 46623589, 'gen': 'testnet-v1.0', 'gh': 'SGO1GKSzyE7IEPItTxCByw9x8FmnrCDexi9/cOUJOiI=', 'grp': 't6YHmDeyzc+ZLDC7YEo
qa5UFeJvsIC/GNyokGJNZrxc=', 'lv': 46624589, 'snd': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'type': 'appl
'}}}
box 650000
```

Balance updated of the user gets updates with correct vaa

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t\xeb\x
10' 32
userBalance 650000
```

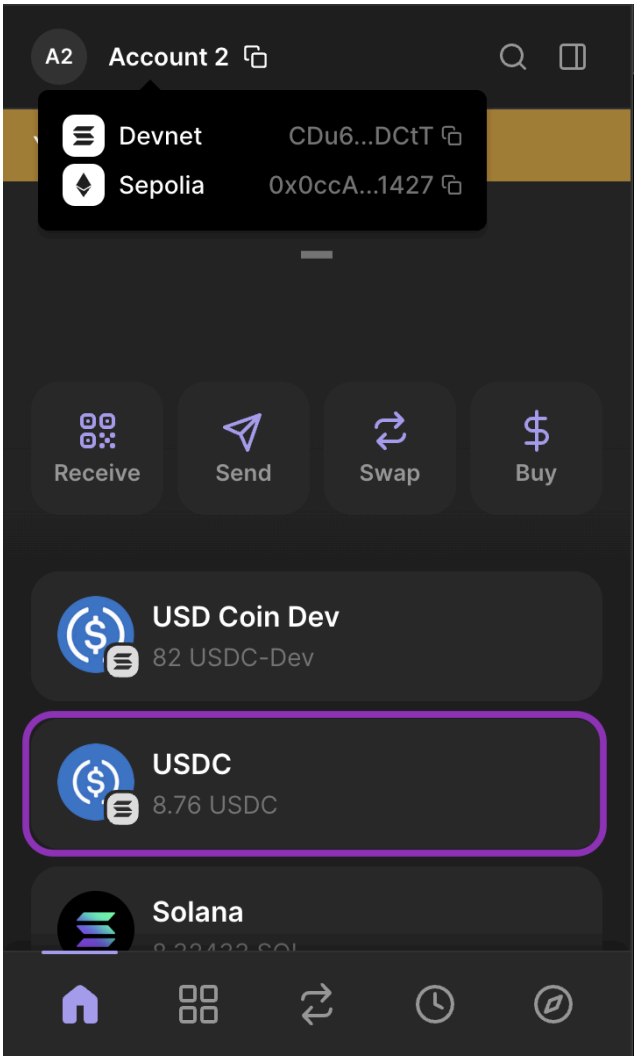
Transaction id on Algorand is — Y52AMOKPADVRNTP5IYZNE7BVPBLKDJVMK2DUWRO7SLPLDOSDF23A

Y52AMOKPA...	46623591	AFIA...64JU →	727096228	Application Call	2 minutes
		CH6M...N2CE →	727096157	Application Call	2 minutes

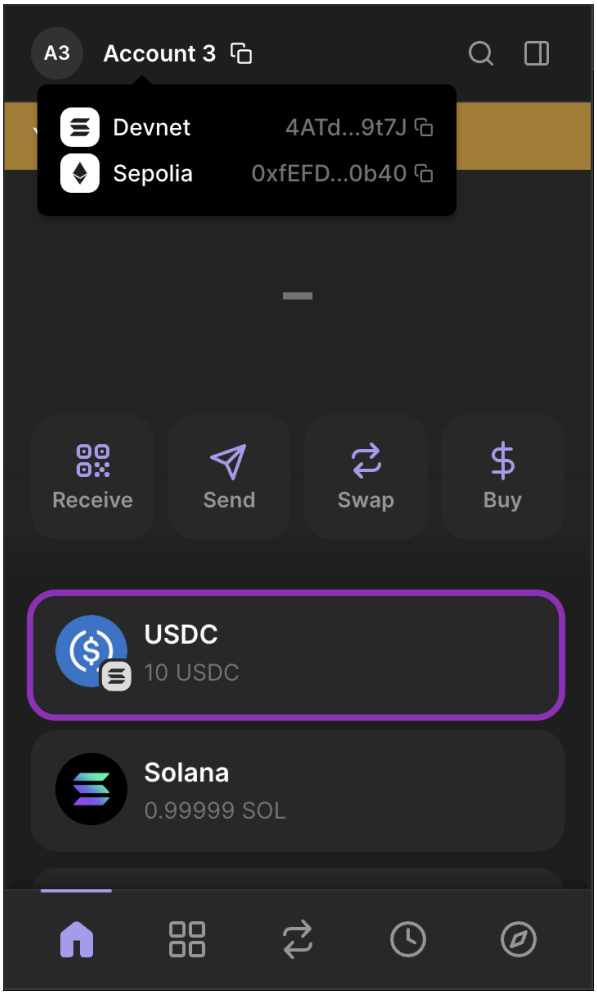
4.5 Manipulating Address in VAA during Deposit

Result	Pass
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect address) can bypass verification and affect user balances, ensuring system resilience against such exploits. The tests focus on two scenarios involving VAA payload manipulation: 1. **Scenario 1:** Modifying the msg.sender in the VAA payload. 2. **Scenario 2:** Modifying both the msg.sender and the login address in the VAA payload. In both scenarios, the aim is to determine if the changes allow the system to process a transaction that updates the balance of an unintended user or rejects it as expected.
Function	Deposit

Steps Performed	
	1. **Check Balance of Both the Users:** • Verify initial token balances of the user1 and user2 in EOA and the codex contract. • User 1 normaliser address: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 == CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT • User 2 hasn't made any deposit transactions so balance in codex will not be shown. • User 2 normaliser address: 2eff66203f97232b945ff476cdcd4f8c1da88d4ee4bbf62e5332e8affd31d4f9 == 4ATdZ1ZsJS5ESTjy1xMfH9Jm27wbZW87KLiPaZ6R9t7J 2. **Deposit Function Execution:** • Execute the deposit function on the Solana chain by user1. • Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** • Retrieve the original VAA from Wormhole scan for the deposit transaction. • Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** • Modify the user1 address in the payload from the correct value a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 to an wrong address value 2eff66203f97232b945ff476cdcd4f8c1da88d4ee4bbf62e5332e8affd31d4f9. • **Scenario 1**: Replace the msg.sender in the payload with a wrong address: • **Wrong Address**: 4ATdZ1ZsJS5ESTjy1xMfH9Jm27wbZW87KLiPaZ6R9t7J. • **Scenario 2**: Replace both the login address and msg.sender with the wrong address. • Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** • Encrypt the manipulated VAA and submit it to the Algorand network manually for verification. • Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** • Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended. • Check if the user's balance is updated correctly. 7. **Final Validation:** • Ensure the manipulated VAA fails and does not alter balances of both user1 and user2. • Confirm the correct VAA succeeds, incrementing the user's 1 balance.

Observation	**Step 1: Check Balance of Both the Users** User Balance of the user1 USDC token = CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 

Balance of the user2 USDC token = 4ATdZ1ZsJS5ESTjy1xMfH9Jm27wbZW87KLiPaZ6R9t7J



User1 Balance in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06\xdd\xd0' 32
userBalance 450000
```

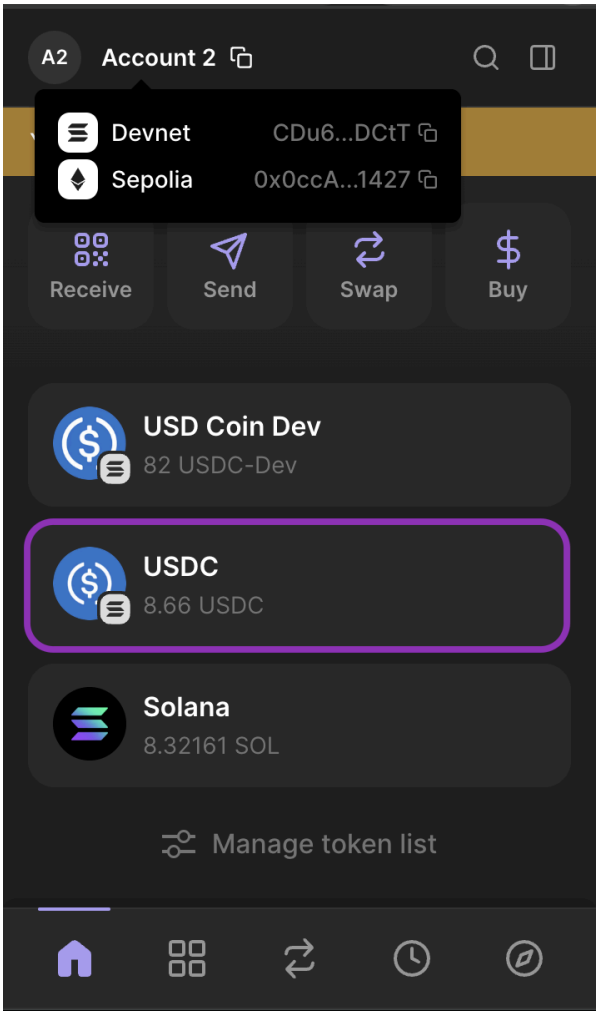
User 2 hasn’t made any deposit transactions so balance in codex will not be shown.

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address 4ATdZ1ZsJS5ESTjy1xMfH9Jm27wbZW87KLiPaZ6R9t7J
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000
Traceback (most recent call last):
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v
2client/algod.py", line 104, in algod_request
    resp = urlopen(req)
           ^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 215, in urlopen
    return opener.open(url, data, timeout)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 521, in open
    response = meth(req, response)
               ^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 630, in http_response
    response = self.parent.error(
               ^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 559, in error
    return self._call_chain(*args)
           ^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 492, in _call_chain
    result = func(*args)
             ^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 639, in http_error_default
    raise HTTPError(req.full_url, code, msg, hdrs, fp)
urllib.error.HTTPError: HTTP Error 404: Not Found

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/scripts/run.py", line 8, in <module>
    readUserBalanceBoxByEnv(SupportedChains.Solana, SupportedChains.Solana, TokenType.USDC)
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/utils/box.py", line 73, in readUserBalanceB
oxByEnv
    return readUserBalanceBox(algod_client, codex_id, loginAddress, loginChainId, assetAddr, assetChainId)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/utils/box.py", line 35, in readUserBalanceB
ox
    box = readBox(algod_client, codex_id, boxKey)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/utils/box.py", line 24, in readBox
    return algod_client.application_box_by_name(app_id, boxKey)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v
2client/algod.py", line 189, in application_box_by_name
    return self.algod_request("GET", req, params=params, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v
2client/algod.py", line 111, in algod_request
    raise error.AlgodHTTPError(e, code)
algosdk.error.AlgodHTTPError: box not found
```

Steps 2: Deposit Function Execution

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run depositUSDC

  > depositUSDC
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --depositUSDC

  secp256k1 unavailable, reverting to browser version
  depositUSDC called
  token account PublicKey [PublicKey(22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87)] {
    _bn: <BN: f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc>
  }


    0 passing (0ms)

  _config PublicKey [PublicKey(3pp7kNQMeGcFQ7BNfF5uWR9pU2rgDjgK7KXGoA5gfwHQ)] {
    _bn: <BN: 29f6f24d8148cb1b0e60424c1a8834046d7a572e82e3713e4d01e57f14af4d6f>
  }
  vaultKey PublicKey [PublicKey(3c4YrWQEqujUEfSdoHyji5te7rtXLWj6Xmi7BUXbyXyi)] {
    _bn: <BN: 26b2b71b78a90614412b39fdc371d659abee46541bad88ad30426502bf4ec671>
  }
  payload <Buffer 64 65 70 6f 73 69 74 a6 be 70 40 7b 6f 5e 03 3e 67 75 ef f7 d2 53 14 cd e5 7c 0d b8 29 2f 5d 54 69 1a 2a 68 9
  3 dc c4 43 43 54 50 55 53 44 43 00 00 00 ... 117 more bytes>
```

USDC Deducted from the wallet



Txn ID —
5qCotZ4m4UtcUt5WtB9rsPnLpiA4v6uLgCUZ5PVpBFK4aj1vSZMPDaV6YddQRygUfFGdRxj5bDvQT97WkFt9zoZ1

Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa

AQAAAAABANsvzxeKxRn3S3w0I9k0hQ7KQdYGQ8SeXPdduJsExvjKf5TP2eyZRfr+bxN9ZGgrk552
dMqxPvViccPQNHRoEB4AZ1kNfAAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkj
AAAAAAAAABMBZGVwb3NpdKa+cEB7b14DPmd17/fSUxTN5XwNuCkvXVRpGipok9zEQ0NUUFVT
REMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABpr5wQHtvXgM+Z
3Xv99JTFM3lfA24KS9dVGkaKmiT3MQ=

Decrypted Raw Vaa

01000000000100db2fcf178ac519f74b7c3423d934850eca41d60643c49e5cf75db89b04c6f8ca7f
94cfd9ec9945fafe6f137d64682b939e7674cab13ef56271c3d0347468101e0067590d7c00000000
0001d18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300000000000
00013016465706f736974a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a689
3dcc4434354505553444300000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000186a0000000000000000
000000000000000000000000000000000000000000000000001a6be70407b6f5e033e6775eff7d
25314cde57c0db8292f5d54691a2a6893dcc4

Decode the raw VAA into a readable format for inspection:-

6465706f736974 :- deposit
a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 :- msg.sender normalizer address
4343545055534443000000000000000000000000000000000000000000000000 :- cctpusdc
000000000000000000000000000000000000000000000000000000000000186a0 :- amount == 100000
0000000000000000000000000000000000000000000000000000000000000001 :- Chain Id
login chain id a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 :- login address normalizer address

Scenario 1: Replace the msg.sender in the payload with a wrong normalizer address

Step 4: Manipulate VAA

a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 == User1 to 2eff66203f97232b945ff476cdcd4f8c1da88d4ee4bbf62e5332e8affd31d4f9 == User2

Wrong decrypted vaa:-

0100000000010057982e6cdea381d1ff0514d526d64742a4f921cffb4c149d7dc068c59b10ef395851
bf9fca8077dec0d5e67c6f08457fede005ad3b12800e9380f20f8ed2081d01675819fd0000000000
01d18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc979230000000000000
0120177697468647261772eff66203f97232b945ff476cdcd4f8c1da88d4ee4bbf62e5332e8affd31d
4f9434354505553444300000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000186a000000000000000
000000000000000000000000000000000000000000000000001a6be70407b6f5e033e6775eff7d253
14cde57c0db8292f5d54691a2a6893dcc4

Step 5: Encrypt and Test Altered VAA

Encrypted wrong vaa:-

AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwh
Ff+3gBa07EoAOk4DyD47SCB0BZ1gZ/QAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v
/yXkjAAAAAAAAABIBd2l0aGRyYXcu/2YgP5cjK5Rf9HbNzU+MHaiNTuS79i5TMuiv/THU+UNDVFBV
U0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAABhqAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAa+cEB7b14DPm
d17/fSUxTN5XwNuCkvXVRpGipok9zE

```
if __name__ == "__main__":
    vaaString = "AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwhFf+3gBa07EoAOk4DyD47SCB0BZ1gZ/
    QAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAAABIBd2l0aGRyYXcu/2YgP5cjK5Rf9HbNzU+MHaiNTuS79i5TMuiv/THU
    +UNDVFBVU0RDAAAAAAAAAAAAAAAAAAA
```

Executing runReceiveMessage.py in codex with wrong vaa and getting an error

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATRQXX2HISIRZDRGXWLRQ46Z6IUJYTIHU3PJGP7PSIQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQ8NAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUONAQ1CDUEQv+BIok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQ8NAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUONAQ1CDUEQv+0Iok=
sig_addr 727096228 R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZK37Y
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bc48b18e53fdaeee77f3473391ac727c63800000000000
```

```
if __name__ == "__main__":
    vaaString = "AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwhFf+3gBa07EoAOk4DyD47SCB0BZ1gZ/
    QAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v/yXkjAAAAAAAAABIBd2l0aGRyYXcu/2YgP5cjK5Rf9HbNzU+MHaiNTuS79i5TMuiv/THU
    +UNDVFBVU0RDAAAAAAAA
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBB
hJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VA
CVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4F
BCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000677561726469616e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x1091cb410>, <algosdk.transaction.Applicat
ionCallTxn object at 0x1091cb350>] ['R7ENA46PT7Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y', 'Z7AISPXE4YFESMU35JXBEM6JZSK
TVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1 434354505553444300000000000000000000000000000000000000000000000 65537 THIS THE
 DATA WE ARE PRINTING
txns [<algosdk.transaction.ApplicationCallTxn object at 0x1091cb410>, <algosdk.transaction.ApplicationCallTxn object at 0x1091
cb350>, <algosdk.transaction.ApplicationCallTxn object at 0x108ec8b90>]
result {'confirmed-round': 46622259, 'pool-error': '', 'txn': {'lsig': {'l': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC
iAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCD
UDNAQlCDUEQv+0Iok='}, 'txn': {'apaa': ['dmVyaWZ5U2lncw==', 'ANsvzxeKxRn3S3w0I9k0hQ7KQdYGQ8SeXPdduJsExvjKf5TP2eyZRfr+bxN9ZGgrk5
52dMqxPvViccPQNHRoEB4A', 'E5R71IsY5T/a7ud/NHM5Gscnxjg=', 'HKEch1q91OBAd819GCnkwNK3DWNhRYYWxuof6IuWh1s='], 'apat': ['R7ENA46PT7
Q5FZJ2QXKWQBM5X6Y4SBJL3EDLOW7OFLCG57PI7363BZKJ7Y', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ'], 'apid': 8652
5623, 'fv': 46622256, 'gen': 'testnet-v1.0', 'gh': 'SGO1GKSzyE7IEPItTxCByw9x8FmnrCDexi9/cOUJOiI=', 'grp': 'kL9ppCqZXjeb5Fdyhdv
50y0j71L79qxUQwODgQt3aOk=', 'lv': 46623256, 'snd': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'type': 'appl
'}}}
box 550000
```

Balance updated of the user gets updates with correct vaa

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08dp'
32
userBalance 550000
```

Transaction id on Algorand is — 7PGG7BOHNKEQJRP4YGLBA6PBGU6GMSS4XQMZF5I7MSX6B2DNSBQA

7PGG7BOHN...	46622259	AFIA...64JU →	727096228	Application Call	2 minutes
		CH6M...N2CE →	727096157	Application Call	2 minutes

4.6 Manipulating “Deposit” String in VAA during Deposit

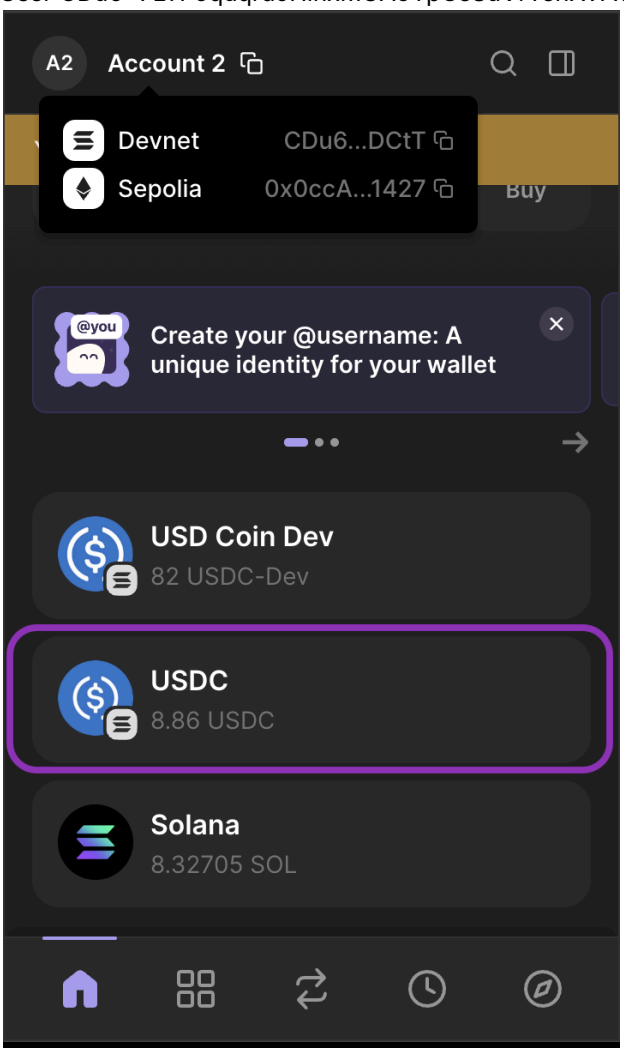
Result	**Pass**
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect deposit string) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Deposit

Steps Performed	1. **Check Balance of the User:** - Verify initial token balances of the user in EOA and the codex contract. 2. **Deposit Function Execution:** - Execute the deposit function on the Solana chain. - Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** - Retrieve the original VAA from Wormhole scan for the deposit transaction. - Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** - Modify the "Deposit" string in the payload from the correct value 6465706f736974 to an invalid value 7769746864726177 "Withdraw" string. - Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** - Encrypt the manipulated VAA and submit it to the Algorand network manually for verification. - Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** - Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended. - Check if the user's balance is updated correctly. 7. **Final Validation:** - Ensure the manipulated VAA fails and does not alter balances. - Confirm the correct VAA succeeds, incrementing the user's balance.

Observation

Step 1: Check the balance of the user in EOA and Codex

User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance



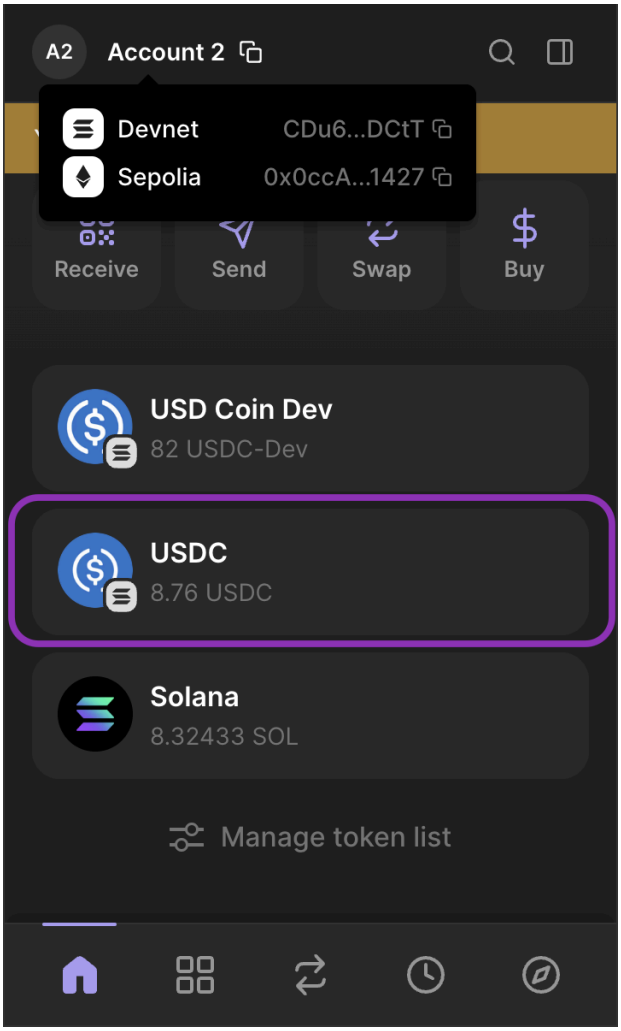
Balance in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x05W0'
32
userBalance 350000
```

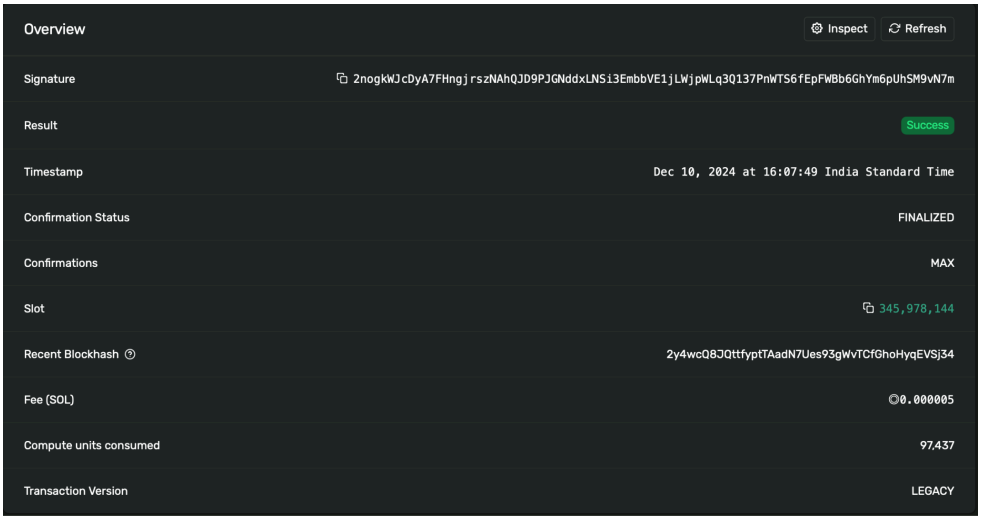
Step 2: Deposit Function Execution

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run depositUSDC

> depositUSDC
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --depositUSDC

secp256k1 unavailable, reverting to browser version
depositUSDC called
token account PublicKey [PublicKey(22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87)] {
  _bn: <BN: f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc>
}


  0 passing (0ms)

_config PublicKey [PublicKey(3pp7kNQMeGcFQ7BNfF5uWR9pU2rgDjgK7KXGoA5gfwHQ)] {
  _bn: <BN: 29f6f24d8148cb1b0e60424c1a8834046d7a572e82e3713e4d01e57f14af4d6f>
}
vaultKey PublicKey [PublicKey(3c4YrWQEqujUEfSdoHyji5te7rtXLWj6Xmi7BUXbyXyi)] {
  _bn: <BN: 26b2b71b78a90614412b39fdc371d659abee46541bad88ad30426502bf4ec671>
}
payload <Buffer 64 65 70 6f 73 69 74 a6 be 70 40 7b 6f 5e 03 3e 67 75 ef f7 d2 53 14 cd e5 7c 0d b8 29 2f 5d 54 69 1a 2a 68 93
 dc c4 43 43 54 50 55 53 44 43 00 00 00 ... 117 more bytes>
```

USDC Deducted from the wallet



Txn ID —
2nogkWJcDyA7FHngjrszNAhQJD9PJGNddxLNSi3EmbbVE1jLWjpWLq3Q137PnWTS6fEpFWBb6GhYm6pUhSM9vN7m



Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa

AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwh
Ff+3gBa07EoAOk4DyD47SCB0BZ1gZ/QAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v
/yXkjAAAAAAAAABIBZGVwb3NpdKa+cEB7b14DPmd17/fSUxTN5XwNuCkvXVRpGipok9zEQ0NUU
FVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABpr5wQHtvXg
M+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQ=

Decrypted Raw Vaa

0100000000010057982e6cdea381d1ff0514d526d64742a4f921cffb4c149d7dc068c59b10ef395851
bf9fca8077dec0d5e67c6f08457fede005ad3b12800e9380f20f8ed2081d01675819fd0000000000
01d18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc979230000000000000
01201
6465706f736974a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc443
4354505553444300000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000186a000000000000000000000000
000000000000000000000000000000000000000000001a6be70407b6f5e033e6775eff7d25314cde
57c0db8292f5d54691a2a6893dcc4

Decode the raw VAA into a readable format for inspection:-

6465706f736974 :- deposit
a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 :- msg.sender normalizer address
4343545055534443000000000000000000000000000000000000000000000000 :- cctpusdc
000000000000000000000000000000000000000000000000000000000000186a0 :- amount == 100000
0000000000000000000000000000000000000000000000000000000000000001 :- Chain Id
login chain id a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4 :- login address normalizer address

Step 4: Manipulate VAA

6465706f736974 == Deposit to 7769746864726177 == withdraw

Wrong decrypted vaa:-

0100000000010057982e6cdea381d1ff0514d526d64742a4f921cffb4c149d7dc068c59b10ef395851
bf9fca8077dec0d5e67c6f08457fede005ad3b12800e9380f20f8ed2081d01675819fd0000000000
01d18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc979230000000000000
01201776974686472617 7a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a689
3dcc4434354505553444300000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000186a000000000000000
000000000000000000000000000000000000000000000000000000000000001a6be70407b6f5e033e6775eff7d
25314cde57c0db8292f5d54691a2a6893dcc4

Step 5: Encrypt and Test Altered VAA

Encrypted wrong vaa:-

AQAAAAABAFeYLmzeo4HR/wUU1SbWR0Kk+SHP+0wUnX3AaMWbEO85WFG/n8qAd97A1eZ8bwh
Ff+3gBa07EoAOk4DyD47SCB0BZ1gZ/QAAAAAAAdGN59MlTckLhwcA9ljoavq4+UxQqyoJRZd8M1v
/yXkjAAAAAAAAABIBd2l0aGRyYXemvnBAe29eAz5nde/30lMUzeV8DbgpL11UaRoqaJPcxENDVFBV
U0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAABhqAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
d17/fSUxTN5XwNuCkvXVRpGipok9zE

```
if __name__ == "__main__":
    vaaString = "AQAAAAABAKmje7Q4B3hjMwTV92zvj8DGuHvhE90KmknnXQCAzusRIo964vR6v5EjqGUDYtX0I1AYq49CxqIKbs4RBIr/
    NPUBZ1BhugAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACQAaGVsbG8AAAAAAAAAAAAAAAAAADwumM9jIsu/
    qi5iP47T0OVF4S0TQAAAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAACcQAM8QEpfqz8RaKq2jKQO6o2WFgMdFm0MoI2EIkQe1mnFbAAAAAAAAAAAAAAAAm00+ZUP1d50hATXDGf+NwvHtl7MA"

    dbEntryID = 86588


    print(b64decode(vaaString).hex(),"This is decrypted raw vaa")
    receiveMessageByEnv(SupportedChains.Solana, SupportedChains.Solana, TokenType.USDC, vaaString, False, dbEntryID)
```

Executing runReceiveMessage.py in codex with wrong vaa and getting an error

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
0100000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a8650362d5f4235018ab8f42c6a20a6ece11048aff34f501675061ba00000000000811
c2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000000240068656c6c6f000000000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d00000000
002712434354555534443000000000000000000000000000000000000000000000000000000000000271200000000000000000000000000000000000000000000000000000000000000000271000cf101297eacfc
2aada32903baa3658580c7459b432823610091077b59a715b000000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300 This is decrypted raw vaa
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxd
VNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNA
CVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 M5OZ7Z3LYC7PG2JSNPDGWMPFMMGBHDATIRLSQTAB2NID5XCGG4RHSRQQBY
Opted in at round: 46685977
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000677561726469616e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x119b6e1e0>, <algosdk.transaction.ApplicationCallTxn object at 0x11a032f30>] ['M5O
Z3LYC7PG2JSNPDGWMPFMMGBHDATIRLSQTAB2NID5XCGG4RHSRQQBY', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1 434354505553444300000000000000000000000000000000000000000000000000 65537 THIS THE DATA WE ARE PRINTING
txns [<algosdk.transaction.ApplicationCallTxn object at 0x119b6e1e0>, <algosdk.transaction.ApplicationCallTxn object at 0x11a032f30>, <algosdk.transaction.Application
llTxn object at 0x1079d8110>]
Traceback (most recent call last):
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v2c
nt/algod.py", line 104, in algod_request
    resp = urlopen(req)
           ^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 215, in urlopen
    return opener.open(url, data, timeout)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 521, in open
    response = meth(req, response)
               ^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 630, in http_response
    response = self.parent.error(
               ^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 559, in error
    return self._call_chain(*args)
           ^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 492, in _call_chain
    result = func(*args)
             ^^^^^^^^^^^
  File "/opt/homebrew/Cellar/python@3.12/3.12.7_1/Frameworks/Python.framework/Versions/3.12/lib/python3.12/urllib/request.py", line 639, in http_error_default
    raise HTTPError(req.full_url, code, msg, hdrs, fp)
urllib.error.HTTPError: HTTP Error 400: Bad Request

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/scripts/runRecieveMessage.py", line 16, in <m
le>
    receiveMessageByEnv(SupportedChains.Solana, SupportedChains.Solana, TokenType.USDC, vaaString, False, dbEntryID)
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/utils/receiveMessage.py", line 298, in receiv
ssageByEnv
    receiveMessageForDeposit(
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/utils/receiveMessage.py", line 373, in receiv
ssageForDeposit
    txid = client.send_transactions( grp )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v2c
nt/algod.py", line 430, in send_transactions
    return self.send_raw_transactn(
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v2c
nt/algod.py", line 358, in send_raw_transaction
    resp = self.algod_request("POST", req, data=txn_bytes, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/codex/venv/lib/python3.12/site-packages/algosdk/v2c
nt/algod.py", line 111, in algod_request
    raise error.AlgodHTTPError(e, code)
algosdk.error.AlgodHTTPError: transaction {_struct:{} Sig:[0 0 0 
```

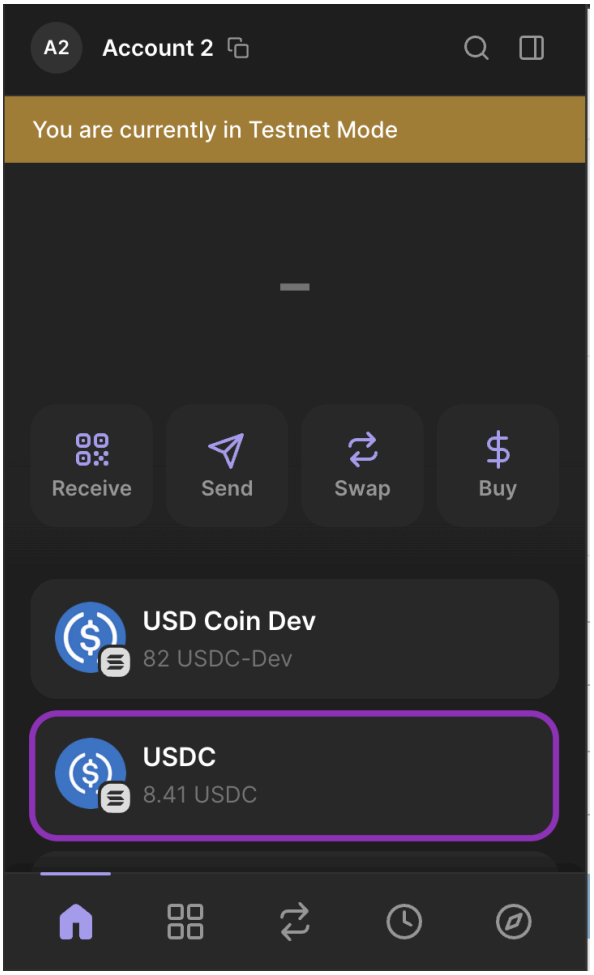
```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
  login chain id 1
  login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
  assest Chain id 65537
  assest address 4343545055534443000000000000000000000000000000000000000000000000
  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06\xdd
  \xd0' 32
  userBalance 450000
```

Transaction id on Algorand is — LVEV7QSBV4ZJHX2UCZ4KF3HS5UZENW3V7BXPIRTESDGU3H6XUIZA

LVEV7QSBV...	46606080	AFIA...64JU →	727096228	Application Call	10 hours, 58 minutes
		CH6M...N2CE →	727096157	Application Call	10 hours, 58 minutes

4.7 Manipulating Token Address in VAA during Withdraw

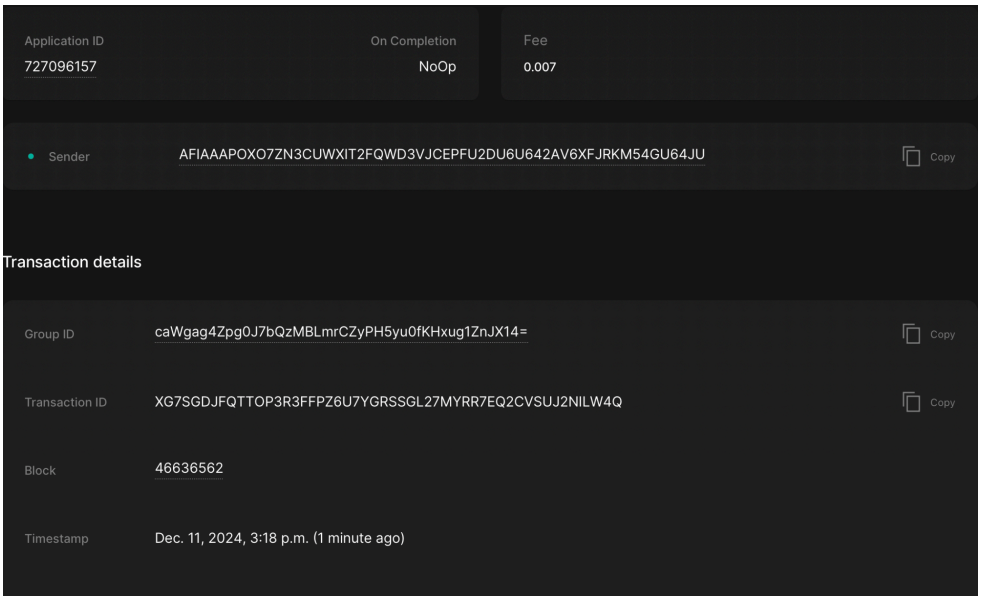
Result	**Pass**
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to Solana, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect address) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw
Steps Performed	1. **Check Balance of the User:** • Verify initial token balances of the user in EOA and the codex contract. 2. **Withdrawal Function Execution:** • Execute the withdrawal function on the Algorand chain. • Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** • Retrieve the original VAA from Wormhole scan for the withdrawal transaction. • Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** • Modify the Chain Id in the payload from the correct value 50000 to a wrong value 20000. • Change from 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc to 0b53375c315f64d1eb58e9ce079b70fa26b7cc2a198561b600a38f565cb769a3. • Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** • Encrypt the manipulated VAA and submit it to the Solana network manually for verification. • Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** • Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. • Check if the user's EOA balance is updated correctly. 7. **Final Validation:** • Ensure the manipulated VAA fails and does not alter balances.

	• Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex** User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance  Balance in codex (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 1 login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT assest Chain id 65537 assest address 4343545055534443000000000000000000000000000000000000000000000000 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08dp' 32 userBalance 550000 **Step 2: Withdrawal Function Execution** (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 727096157 before algo amount of codex 2000000 codex min balance 409100 before algo amount of msg processor 2000000 1 CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT dest_private_key TtNrm5b1TvgbW5HhkyifVLeRoifvk6LX64c3V4ffv19Zky7F3gWJ7FxEoUkM6JCkkJ5rp7DWPXDXZAQTCaY6aEP 4343545055534443000000000000000000000000000000000000000000000000 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 1 emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL KVNGXKQ4OCKMU5Y login CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1 super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 1 236 1 b'TX{"Login_Address": "CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT", "Login_ChainId": "1", "Token_Address": "4343545055534443 0000000000000000000000000000000000000000000000", "Receiver_Address": "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87", "Receiv er_ChainId": "1", "Amount": "50000", "Withdraw_Fee": "False"}pr5wQHtvXqM+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQAAAAAAAAAAUNDVFBVU0RDA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADzxpeFz1EYlzZf4LHDc//FG/NQdtAg7wL3NHMWYIAvwAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMN QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' userBalance 500000 after algo amount of codex 2000000 after algo amount of msg processor 2000000

Balance after withdrawal

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
 ' 32
userBalance 500000
```

Txn Id:- XG7SGDJFQTTOP3R3FFPZ6U7YGRSSGL27MYRR7EQ2CVSUJ2NILW4Q



Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa:-
AQAAAAABABWDYOlQnIF2BFI3wlDDQrQixs6FUQjast/0G3ii5br5ffDtf5X7kjT6jKmBERTfmMrIUT1Cw9/XW0D8dye1/ucBZ1mtUQAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAC0Ad2l0aGRyYXcPPGI4XPURiXNI/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQALm/Iw0lhObn7jspX59T+DRIIy9fZiMfkhoVZUTpqF250Y3n0yVNyQuHBwD2WOhq+rj5TFCrKgIFI3wzW//JeSMA

Decrypted Raw Vaa:-
010000000000100158360e9509c8176045237c250c342b422c6ce855108dab2dff41b78a2e5baf97df0ed7f95fb9234fa8ca9811114df98cac8513d42c3dfd75b40fc7727b5fee7016759ad510000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002d0077697468647261770f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc00000000000000014343545055534443000000000000000000000000000000000000000000000000

Transaction id :-
88c238e94b0ff2b590d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 4: Manipulate VAA

0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc == 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 to 0b53375c315f64d1eb58e9ce079b70fa26b7cc2a198561b600a38f565cb769a3 == mD5jLDsxRciyoaFncWxy4RXfoJMQnA9Xmbwf2hmWsEa

22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 owner is User1 which is CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
mD5jLDsxRciyoaFncWxy4RXfoJMQnA9Xmbwf2hmWsEa owner is User2 which is 4ATdZ1ZsJS5ESTjy1xMfH9Jm27wbZW87KLiPaZ6R9t7J

Wrong decrypted vaa:-
010000000001001583 60e9509c8176045237c250c342b422c6ce855108dab2dff41b78a2e5baf97df0ed7f95fb9234fa8ca9811114df98cac8513d42c3dfd75b40fc7727b5fee7016759ad510000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002d007769746864726177 0b53375c315f64d1eb58e9ce079b70fa26b7cc2a198561b600a38f565cb769a300000000000000014343545055534443000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000c35000b9bf230d2584e6e7ee3b295f9f53f83465232f5f66231f921a156544e9a85db9d18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 5: Encrypt and Test Altered VAA

Encrypted wrong vaa:-
AQAAAAABABWDYOlQnIF2BFI3wlDDQrQixs6FUQjast/0G3ii5br5ffDtf5X7kjT6jKmBERTfmMrIUT1Cw9/XW0D8dye1/ucBZ1mtUQAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAAC0Ad2l0aGRyYXcLUzdcMV9k0etY6c4Hm3D6JrfMKhmFYbYAo49WXLdpowAAAAAAAABQ0NUUFVTREMAAAAAAAAA

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (0ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 15 83 60 e9 50 9c 81 76 04 52 37 c2 50 c3 42 b4 22 c6 ce 85 51 08 da b2 df f4 1b 78 a2
  e5 ba f9 7d f0 ed 7f 95 fb 92 34 fa 8c a9 ... 259 more bytes>
  SendTransactionError: failed to send transaction: Transaction precompile verification failure InvalidAccountIndex
      at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
  DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
      at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
  dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
    logs: undefined
  }
```

Step 6: Re-test with Correct VAA

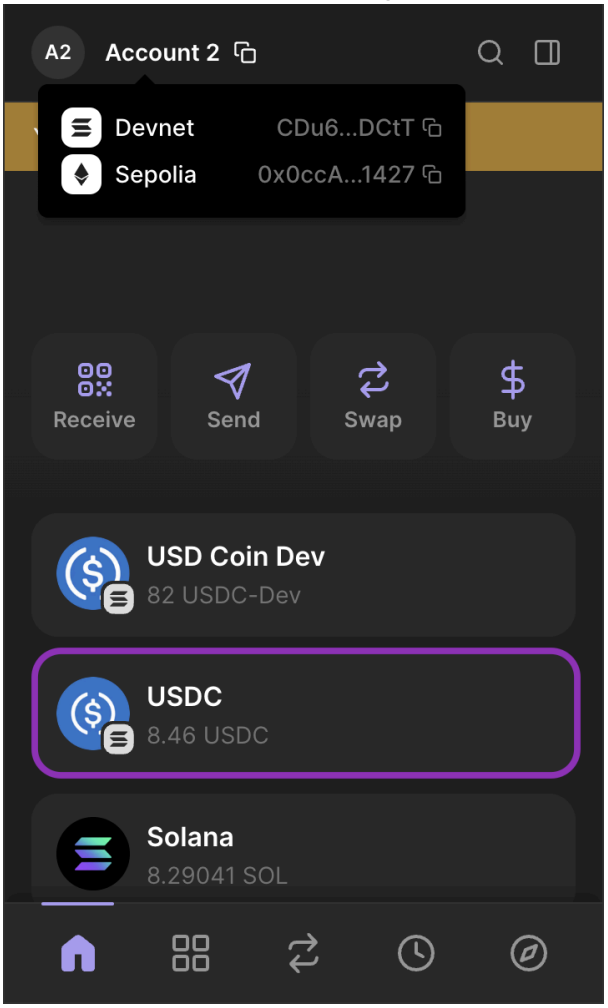
Entering the correct vaa

```
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAAABAWDYOlQnIF2BFI3wlDDQrQixs6FUQjast/0G3ii5br5ffDtf5X7kjT6jKmBERTfmMrIUT1Cw9/XW0D8dye1/
  ucBZ1mtUQAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAAC0Ad2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQALm/
  Iw0lhObn7jspX59T+DRlIy9fZiMfkhoVZUTpqF250Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing Withdraw in solana with correct vaa

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (0ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 15 83 60 e9 50 9c 81 76 04 52 37 c2 50 c3 42 b4 22 c6 ce 85 51 08 da b2 df f4 1b 78 a2
  e5 ba f9 7d f0 ed 7f 95 fb 92 34 fa 8c a9 ... 259 more bytes>
  response [
    {
      signature: '2CDfwWsum5zqHZiAERjHEWYXqWLYmWMt4EfHCyxPsVHqWCUJ6SynPfkn3sW3WAp6SBrH5XSSqcU61VCKxH3oh3ke',
      response: { context: [Object], value: [Object] }
    },
    {
      signature: 'GKDCPZLqGdXBqD5JupYFS6kjNHv5MXyRFJ6EMpADGfXR1jm3MArri9Xejv3bc6dbmsqdxpaRD8fn5XiBWkXoH3a',
      response: { context: [Object], value: [Object] }
    }
  ]
  createWithdrawTokenInstruction is called
  token PublicKey [PublicKey(4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU)] {
    _bn: <BN: 3b442cb3912157f13a933d0134282d032b5ffecd01a2dbf1b7790608df002ea7>
  }
  receivedAcc PublicKey [PublicKey(2GmgKjhg6P5ppxKs4QFmgWwAj3tpB4FhNfdfY6nzxaif)] {
    _bn: <BN: 12e5eaa7916934baa0cf72f712e7084ea4c420dfd806f24c3596d2e2afbb821c>
  }
  posted PublicKey [PublicKey(AzAkTMNYtwieL7RwYGL16a1ek9F82MZsxD4sj5BrRGEf)] {
    _bn: <BN: 945e4fda454508dd418826982a2b6fa39554dbf62c7e909654bf4e4a60f7ce44>
  }
```

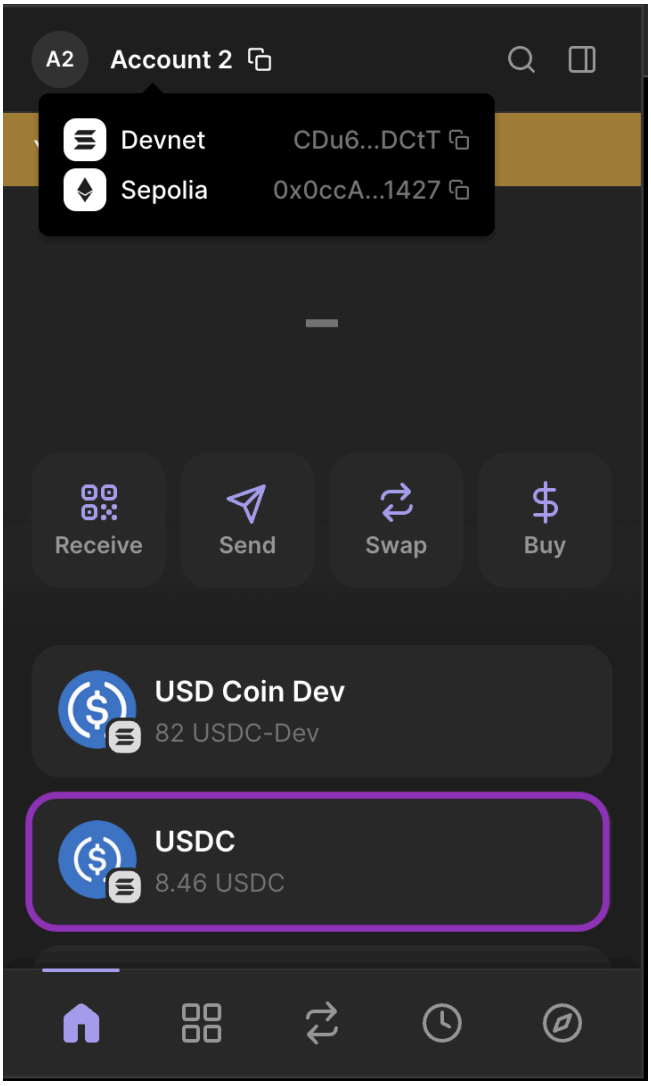
Balance updated of the user EOA gets updates with correct vaa



	Transaction id on Solana is:- 2yKA4N9pzxHw2fFvH1f9Cid1N2C9dBJLABZXRSmsV3CqEnQ93xuzWYzFbhiZaKTMBkQgFQzHQw2VEZZEifij2RKi Overview Inspect Refresh Signature 2yKA4N9pzxHw2fFvH1f9Cid1N2C9dBJLABZXRSmsV3CqEnQ93xuzWYzFbhiZaKTMBkQgFQzHQw2VEZZEifij2RKi Result Success Timestamp Dec 11, 2024 at 21:04:01 India Standard Time Confirmation Status FINALIZED Confirmations MAX Slot 346,256,820 Recent Blockhash 88ZUAecUyPDtZen2hmG4u3CajcW9y9U6WCMFjgDZ3qNY Fee (SOL) ◎0.000005 Compute units consumed 57,696 Transaction Version LEGACY

4.8 Manipulating "Withdraw" String in VAA during Withdraw

Result	**Pass**
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to Solana, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect Withdraw string) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw
Steps Performed	1. **Check Balance of the User:** • Verify initial token balances of the user in EOA and the codex contract. 2. **Withdrawal Function Execution:** • Execute the withdrawal function on the Algorand chain. • Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** • Retrieve the original VAA from Wormhole scan for the withdrawal transaction. • Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** • Modify the Chain Id in the payload from the correct value Withdraw string 7769746864726177 to a wrong value "hello" String 68656c6c6f000000 . • Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** • Encrypt the manipulated VAA and submit it to the Solana network manually for verification. • Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** • Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. • Check if the user's EOA balance is updated correctly.

	7. **Final Validation:** ● Ensure the manipulated VAA fails and does not alter balances. ● Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex** User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance  Balance in codex (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 1 login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT assest Chain id 65537 assest address 4343545055534443000000000000000000000000000000000000000000000000 b'\x00\x00\x00

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
1 CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
dest_private_key TtNrm5b1TvgbW5HhkyifVLeRoifvk6LX64c3V4ffv19Zky7F3gWJ7FxEoUkM6JCkkJ5rp7DWPXDXZAQTCaY6aEP
4343545055534443000000000000000000000000000000000000000000000000 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 1
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 1
236
1
b'TX{"Login_Address": "CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT", "Login_ChainId": "1", "Token_Address": "4343545055534443
0000000000000000000000000000000000000000000000", "Receiver_Address": "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87", "Receiv
er_ChainId": "1", "Amount": "50000", "Withdraw_Fee": "False"}pr5wQHtvXgM+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQAAAAAAAAAAUNDVFBVU0RDA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAADzxpeFz1EYlzZf4LHDc//FG/NQdtAg7wL3NHMWYIAvwAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMN
QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 700000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Balance after withdrawal

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\n\xae`'
 32
userBalance 700000
```

Txn Id:- RDBDR2KLB7ZLLEGRGM2FIP4KIEPENWWOD4FXJVM6WGP2M72XOEXQ

Application ID	On Completion	Fee
727096157	NoOp	0.007

Sender AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU Copy

Transaction details

Group ID	OgjB5HUbQ4HLwaDxdunb6Mr0S4/89bf7JEEE7o1RwM4=	Copy
Transaction ID	RDBDR2KLB7ZLLEGRGM2FIP4KIEPENWWOD4FXJVM6WGP2M72XOEXQ	Copy
Block	46626536	
Timestamp	Dec. 11, 2024, 7:40 a.m. (14 minutes ago)	
Transaction type	Application Call	

Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa:-

AQAAAAABAMktuuSV8DkR3qkno8YBXvi2aqq4dg48KLwQEYf9ZLN0Ie5MYXvtyPv/oUQ48e/m7bP
Yx1za9b6a/UcIkICJuLIAZ1lB0wAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLA
AAAAAAAACkAd2I0aGRyYXcPPGI4XPURiXNI/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAAAB
Q0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAIjCOOILD/K1kNEzNFQ/ikEeRtrOHwt01Z6xn6Z/V3Ev
0Y3n0yVNyQuHBwD2WOhq+rj5TFCrKgIFI3wzW//JeSMA

Decrypted Raw Vaa:-

01000000000100c92dbae495f03911dea927a3c6015ef8b66aaab8760e3c28bc101187fd64b37421e
e4c617bedc8fbffa14438f1efe6edb3d8c75cdaf5be9afd4708908089b8b200675941d3000000000
00811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000
0290077697468647261770f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f73473166080
2fc0000000000000001434354505553444300000000000000000000000000000000000000000
00000000 0000000000000001

0000000000000000000000000000000000000000000000000000000000000000c3500088c238e94b0ff2b590d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Decode the raw VAA into a readable format for inspection:-
Withdraw string :- 7769746864726177
Token address :- 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc
Destination chain id :- 0000000000000001
Token address unified USDC :-
4343545055534443000000000000000000000000000000000000000000000000
Destination chain id :- 0000000000000001
Amount :- 0000000000000000000000000000000000000000000000000000000000000000c350 == 50000
Native bool :- 00
Transaction id :-
88c238e94b0ff2b590d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 4: Manipulate VAA

7769746864726177 == withdraw to
68656c6c6f000000 == hello

Wrong decrypted vaa:-
01000000000100c92dbae495f03911dea927a3c6015ef8b66aaab8760e3c28bc101187fd64b37421ee4c617bedc8fbffa14438f1efe6edb3d8c75cdaf5be9afd4708908089b8b200675941d300000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000000290068656c6c6f0000000f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc0000000000000000143435450555344430000000000000000000000000000000000000000000000000000000000

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called
  <Buffer 01 00 00 00 00 01 00 c9 2d ba e4 95 f0 39 11 de a9 27 a3 c6 01 5e f8 b6 6a aa b8 76 0e 3c 28 bc 10 11 87 fd 64 b3 74 2
  1 ee 4c 61 7b ed c8 fb ff a1 44 ... 259 more bytes> This is the raw vaa
  0100000000100c92dbae495f03911dea927a3c6015ef8b66aaab8760e3c28bc101187fd64b37421ee4c617bedc8fbffa14438f1efe6edb3d8c75cdaf5be9a
  fd4708908089b8b200675941d300000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000000290068656c
  6c6f0000000f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc000000000000000143435450555344430000000000000000000
  0000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000c3500088c238e94b0ff2b5
  90d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300 This is the
   decoded message from algorand vaa


    0 passing (1ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 c9 2d ba e4 95 f0 39 11 de a9 27 a3 c6 01 5e f8 b6 6a aa b8 76 0e 3c 28 bc 10 11 87 fd
  64 b3 74 21 ee 4c 61 7b ed c8 fb ff a1 44 ... 259 more bytes>
  SendTransactionError: failed to send transaction: Transaction precompile verification failure InvalidAccountIndex
      at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
  DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
      at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
  dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
    logs: undefined
  }
```

Step 6: Re-test with Correct VAA

Entering the correct vaa

```
> withdraw                                                    4 of 5
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAMktuuSV8DkR3qkno8YBXvi2aqq4dg48KLwQEYf9ZLN0Ie5MYXvtyPv/oUQ48e/m7bPYx1za9b6a/
  UcIkICJuLIAZ1lB0wAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAACkAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAIjCOOlLD/K1kNEzNFQ/ikEeRtr0Hwt01Z6xn6Z/
  V3Ev0Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  console.log(signedMsg,"This is the raw vaa")
  console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing Withdraw in solana with correct vaa

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called
  <Buffer 01 00 00 00 00 01 00 c9 2d ba e4 95 f0 39 11 de a9 27 a3 c6 01 5e f8 b6 6a aa b8 76 0e 3c 28 bc 10 11 87 fd 64 b3 74 21 ee 4c 61 7b ed c8 fb ff a1 44 ... 2
  59 more bytes> This is the raw vaa
  01000000000100c92dbae495f03911dea927a3c6015ef8b66aaab8760e3c28bc101187fd64b37421ee4c617bedc8fbffa14438f1efe6edb3d8c75cdaf5be9afd4708908089b8b200675941d300000000000
  811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000029007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc
  0000000000000001434354505553444300000000000000
```

	Transaction id on Solana is — 3VkbeHv9fZGaZtLrfBBM3eYvJJDh7B1gPX8bYpX92hgYPeafvD91ADwuTMv6jKP9AW6VfYzyxVnVWLRdoxGwrBLq Overview Inspect Refresh Signature 3VkbeHv9fZGaZtLrfBBM3eYvJJDh7B1gPX8bYpX92hgYPeafvD91ADwuTMv6jKP9AW6VfYzyxVnVWLRdoxGwrBLq Result Success Timestamp Dec 11, 2024 at 14:26:04 India Standard Time Confirmation Status FINALIZED Confirmations MAX Slot 346,193,002 Recent Blockhash D8P3ey3S6b9Aa1pz7dHYADt67U6xz4w4jcS7o9gg5HJJ Fee (SOL) ◎0.000005 Compute units consumed 60,696 Transaction Version LEGACY

4.9 Manipulating Amount in VAA during Withdraw

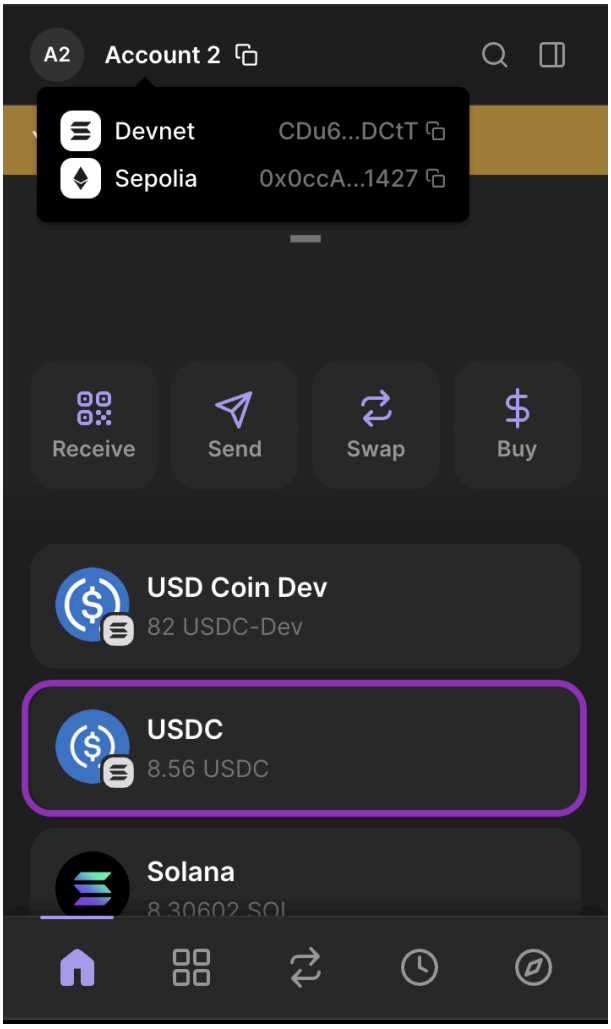
Result	**Pass**
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to Solana, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect amount) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw

Steps Performed	
	1. **Check Balance of the User:** - Verify initial token balances of the user in EOA and the codex contract. 2. **Withdrawal Function Execution:** - Execute the withdrawal function on the Algorand chain. - Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** - Retrieve the original VAA from Wormhole scan for the withdrawal transaction. - Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** - Modify the Chain Id in the payload from the correct value 50000 to a wrong value 20000. - Change from 0000000000000000000000000000000000000000000000000000000000000000c350 to 0000000000000000000000000000000000000000000000000000000000000000004e20. - Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** - Encrypt the manipulated VAA and submit it to the Solana network manually for verification. - Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** - Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. - Check if the user's EOA balance is updated correctly. 7. **Final Validation:** - Ensure the manipulated VAA fails and does not alter balances. - Confirm the correct VAA succeeds, incrementing the user's balance.

Observation

Step 1: Check the balance of the user in EOA and Codex

User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance



Balance in codex

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t'\xc0"
 32
userBalance 600000
```

Step 2: Withdrawal Function Execution

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
1 CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
dest_private_key TtNrm5b1TvgbW5HhkyifVLeRoifvk6LX64c3V4ffv19Zky7F3gWJ7FxEoUkM6JCkkJ5rp7DWPXDXZAQTCaY6aEP
4343545055534443000000000000000000000000000000000000000000000000 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 1
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 1
236
1
b'TX{"Login_Address": "CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT", "Login_ChainId": "1", "Token_Address": "4343545055534443
000000000000000000000000000000000000000000000000", "Receiver_Address": "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87", "Receiv
er_ChainId": "1", "Amount": "50000", "Withdraw_Fee": "False"}pr5wQHtvXgM+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQAAAAAAAAAAUNDVFBVU0RDA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADzxpeFz1EYlzZf4LHDc//FG/NQdtAg7wL3NHMWYIAvwAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMN
QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 550000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Balance after withdrawal

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08dp'
32
userBalance 550000
```

Txn Id:- VDHF66XDSM4NTHT7MVX3OJUC3PBLL6IX23CTLXGCHUQXK6NNC45A

Application ID: 727096157
On Completion: NoOp
Fee: 0.007

Sender: AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU

Transaction details

Group ID: Ks1Cga1uWf9atFfn0bHYLMP7I4NBfATaDd5myTiza/4=

Transaction ID: VDHF66XDSM4NTHT7MVX3OJUC3PBLL6IX23CTLXGCHUQXK6NNC45A

Block: 46629880

Timestamp: Dec. 11, 2024, 10:13 a.m. (0 minutes ago)

Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa:-

AQAAAAABAJxfAibzqJmVXmkChMSMEpJ1mAdiB6/OFzYdoxUY3Cv7YKSrLzamxsjNB2+dIW73HYrqOwG+7Yr6L+GtXUG0GZwBZ1llswAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACwAd2I0aGRyYXcPPGI4XPURiXNI/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAKjOX3rjkzjZnn9Ib7cmgtvCtfkX1sU13MI9IXV5rRc60Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFI3wzW//JeSMA

Decrypted Raw Vaa:-

010000000001009c5f0226f3a899955e690284c48c12927598076207afce17361da31518dc2bfb60a4ab2f36a6c6c8cd076f9d216ef71d8aea3b01beed8afa2fe1ad5d41b4199c01675965b300000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002c007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f73473166080 2fc0000000000000001434354505553444300000000000000000000000000000000000000000000000000

Transaction id :-
88c238e94b0ff2b590d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 4: Manipulate VAA
0000000000000000000000000000000000000000000000000000000000000000c350 == 50000 to
0000000000000000000000000000000000000000000000000000000000000000004e20 == 20000

Wrong decrypted vaa:-
010000000001009c5f0226f3a899955e690284c48c12927598076207afce17361da31518dc2bfb60a4ab2f36a6c6c8cd076f9d216ef71d8aea3b01beed8afa2fe1ad5d41b4199c01675965b300000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002c007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f73473166080 2fc0000000000000014343545055534443000000000000000000000000000000000000000000000000000000000

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

> withdraw
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

secp256k1 unavailable, reverting to browser version
withdraw called


  0 passing (1ms)

wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
wallet Uint8Array(64) [
   23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
  225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
  226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
  190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
  247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
   93,  84, 105,  26,  42, 104, 147, 220, 196
]
wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
signedMsg <Buffer 01 00 00 00 00 01 00 9c 5f 02 26 f3 a8 99 95 5e 69 02 84 c4 8c 12 92 75 98 07 62 07 af ce 17 36 1d a3 15 18
dc 2b fb 60 a4 ab 2f 36 a6 c6 c8 cd 07 6f ... 259 more bytes>
SendTransactionError: failed to send transaction: Transaction precompile verification failure InvalidAccountIndex
    at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
  logs: undefined
}
```

Step 6: Re-test with Correct VAA

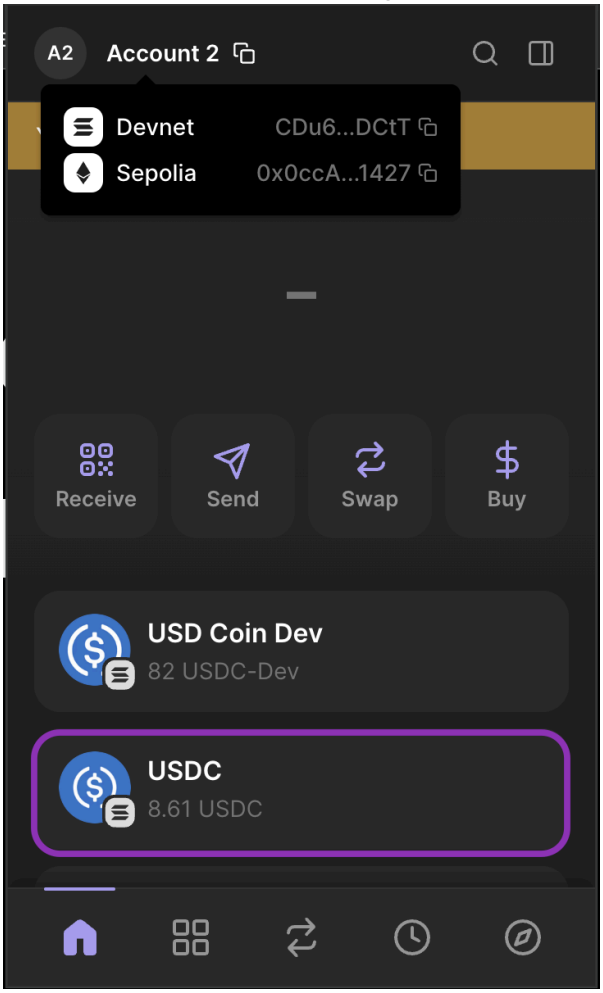
Entering the correct vaa

```
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAJxfAibzqJmVXmkChMSMEpJ1mAdiB6/OFzYdoxUY3Cv7YKSrLzamxsjNB2+dIW73HYrqOwG+7Yr6L
  +GtXUG0GZwBZ1llswAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACwAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAKjOX3rjkzj
  Znn9lb7cmgtvCtfkX1sU13MI9IXV5rRc60Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing Withdraw in solana with correct vaa

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (0ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 9c 5f 02 26 f3 a8 99 95 5e 69 02 84 c4 8c 12 92 75 98 07 62 07 af ce 17 36 1d a3 15 18
  dc 2b fb 60 a4 ab 2f 36 a6 c6 c8 cd 07 6f ... 259 more bytes>
  response [
    {
      signature: '63H8eGvTTy8Dt6UYkRZGBe86ZFkZAsmX8iTbBJksLNoYcwedDU2bNaZXLEwdjXXLvsXJvKEXNG9WuVLjd47JQxmo',
      response: { context: [Object], value: [Object] }
    },
    {
      signature: '5JPXW2B83JWTiFRiysekBhygFkJMkFueuzDf8qcthYCegXvo6VVRKw2iZkceVtFkzv5FHZJ8boSNwLA4SFU6gQVj',
      response: { context: [Object], value: [Object] }
    }
  ]
  createWithdrawTokenInstruction is called
  token PublicKey [PublicKey(4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU)] {
    _bn: <BN: 3b442cb3912157f13a933d0134282d032b5ffecd01a2dbf1b7790608df002ea7>
  }
  receivedAcc PublicKey [PublicKey(52MEs9viERbxSmyH91Wu1He2bQSiEJRzEPeQeYtAdgej)] {
    _bn: <BN: 3bc7367f290dc373421b0c50a997198f02c3af78b2c1c2b9df6346323b9cca58>
  }
  posted PublicKey [PublicKey(B8d3wQj2mqEzokyrAghyZ8bd1WbwK7MNQDthNw83Ltj7)] {
    _bn: <BN: 9688b2919f0142c960d8065b48860b8c71fcf8f96af969d2333c67fd910ea62e>
  }
```

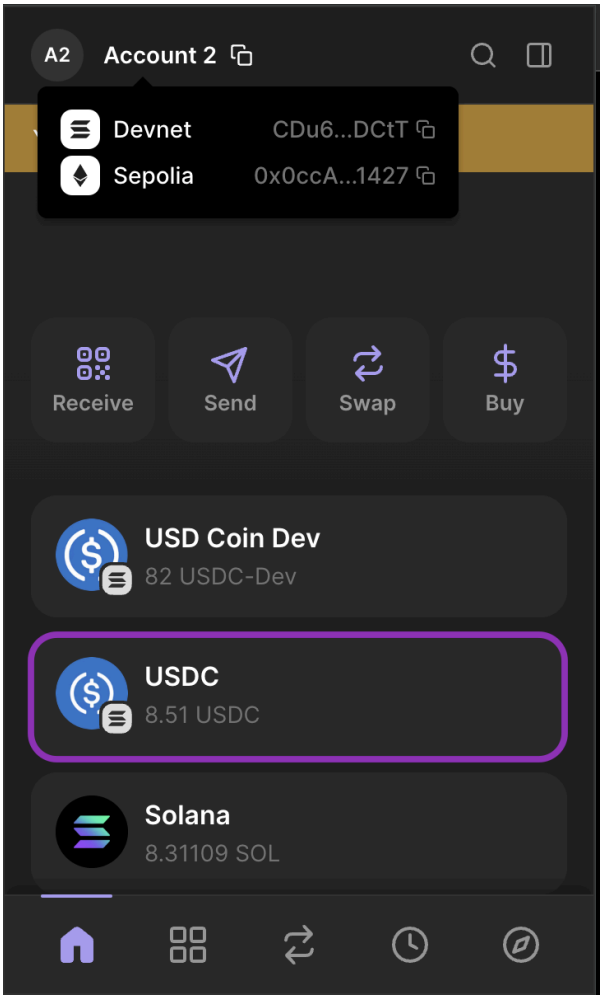
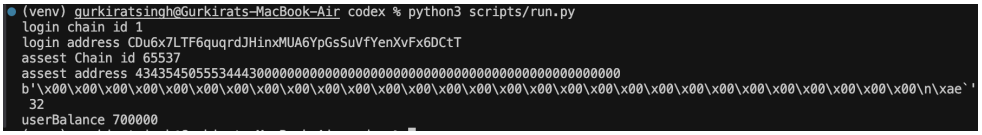
Balance updated of the user EOA gets updates with correct vaa



	Transaction id on Solana is — 5fGbuqdsMct9HmRpY2spsJmmg6fKEJ3UfUfZY8SXL2j1fZB7UUzbFRcWCLN4hCsWB6mAo1Rd6w1GsLDj8EhsqeZr Overview Inspect Refresh Signature 5fGbuqdsMct9HmRpY2spsJmmg6fKEJ3UfUfZY8SXL2j1fZB7UUzbFRcWCLN4hCsWB6mAo1Rd6w1GsLDj8EhsqeZr Result Success Timestamp Dec 11, 2024 at 16:01:59 India Standard Time Confirmation Status FINALIZED Confirmations MAX Slot 346,208,372 Recent Blockhash 4DjXAaEuhee1XHqEWWRHWSCBDGnLbaryyQHV8crrGSHj Fee (SOL) ◎0.000005 Compute units consumed 62,196 Transaction Version LEGACY

4.10 Manipulating ChainId in VAA during Withdraw

Result	**Pass**
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to Solana, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect chain id) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw
Steps Performed	1. **Check Balance of the User:** • Verify initial token balances of the user in EOA and the codex contract. 2. **Withdrawal Function Execution:** • Execute the withdrawal function on the Algorand chain. • Capture the transaction hash for further analysis. 3. **Collection And Decrypt Of Correct Raw VAA:** • Retrieve the original VAA from Wormhole scan for the withdrawal transaction. • Decode the raw VAA into a readable format for inspection. 4. **Manipulate VAA:** • Modify the Chain Id in the payload from the correct value 1 to a wrong value 2. • **Scenario 1:** Change the chain ID in **one location** within the VAA payload. • Change 0000000000000001 to 0000000000000002. • **Scenario 2:** Change the chain ID in **two locations** to simulate a more complex manipulation. • Reconstruct the decrypted VAA with the modified payload. 5. **Encrypt and Test Altered VAA:** • Encrypt the manipulated VAA and submit it to the Solana network manually for verification. • Confirm that the transaction is rejected, ensuring no balance update occurs. 6. **Re-test with Correct VAA:** • Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. • Check if the user's EOA balance is updated correctly. 7. **Final Validation:**

	• Ensure the manipulated VAA fails and does not alter balances. • Confirm the correct VAA succeeds, incrementing the user's balance.
Observation	**Step 1: Check the balance of the user in EOA and Codex** User CDu6×7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT USDC Balance  Balance in codex  (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 1 login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT assest Chain id 65537 assest address 4343545055534443000000000000000000000000000000000000000000000000 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\n\xae``' 32 userBalance 700000 **Step 2: Withdrawal Function Execution**

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
1 CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
dest_private_key TtNrm5b1TvgbW5HhkyifVLeRoifvk6LX64c3V4ffv19Zky7F3gWJ7FxEoUkM6JCkkJ5rp7DWPXDXZAQTCaY6aEP
4343545055534443000000000000000000000000000000000000000000000000 22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87 1
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWR
IQLKVNGXKQ4OCKMU5Y
login CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT 1
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 1
236
1
b'TX{"Login_Address": "CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT", "Login_ChainId": "1", "Token_Address": "4343545055534
4430000000000000000000000000000000000000000000000", "Receiver_Address": "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87", "
Receiver_ChainId": "1", "Amount": "50000", "Withdraw_Fee": "False"}pr5wQHtvXgM+Z3Xv99JTFM3lfA24KS9dVGkaKmiT3MQAAAAAAAAAAUND
VFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADzxpeFz1EYlzZf4LHDc//FG/NQdtAg7wL3NHMWYIAvwAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAMNQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 650000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Balance after withdrawal

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 1
login address CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t\xe
b\x10' 32
userBalance 650000
```

Txn Id:- KIWKZ2NDTN7JXIZKSY3TTODXNRBSPJZA6IQT32FXUH6M43W52Q6Q

Application ID	On Completion	Fee
727096157	NoOp	0.007

Sender: AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU (Copy)

Transaction details

Group ID	WAzdSAUU4LeI6r+O6WTXAAdQt3EYcvUCrcUAeiB8ppk= (Copy)
Transaction ID	KIWKZ2NDTN7JXIZKSY3TTODXNRBSPJZA6IQT32FXUH6M43W52Q6Q (Copy)
Block	46628768
Timestamp	Dec. 11, 2024, 9:22 a.m. (1 minute ago)
Transaction type	Application Call

Step 3: Collection And Decrypt Of Correct Raw VAA

RAW Vaa:-

AQAAAAABAJXuUPuxx71ApdAoZk+qnn37qiUuN2ZDdfmSbbFfyxNwKrkazQEv00u7j513FK58tyuSA
J5AwjZB1w/LyHKuxT8BZ1lZ0AAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAA
AAAAAAACoAd2I0aGRyYXcPPGI4XPURiXNI/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAAABQ
0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAFIsrOmjm36boyqWNzm4d2xDJ6cg8iE96Leh/M5u3d
Q90Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFI3wzW//JeSMA

Decrypted Raw Vaa:-

0100000000010095ee50fbb1c7bd40a5d028664faa9e7dfbaa252e37664375f9926db15fcb13702a
b91acd012fd34bbb8f9d7714ae7cb72b92009e40c23641d70fcbc872aec53f01675959d000000000
000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
002a007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f7347316608

02fc0000000000000014343545055534443000000000000000000000000000000000000000
0000000000000000000000010000000000000000000000000000000000000000000000000
00000000000c35000522cace9a39b7e9ba32a963739b8776c4327a720f2213de8b7a1fcce6eddd4
3dd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Decode the raw VAA into a readable format for inspection:-
Withdraw string :- 7769746864726177
Token address :- 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f734731660802fc
Destination chain id :- 0000000000000001
Token address unified USDC :-
434354505553444300000000000000000000000000000000000000000000000000
Destination chain id :- 0000000000000001
Amount :- 0000000000000000000000000000000000000000000000000000000000000c350
== 50000
Native bool :- 00
Transaction id :-
88c238e94b0ff2b590d13334543f8a411e46dace1f0b74d59eb19fa67f57712fd18de7d3254dc90b87
0700f658e86afab8f94c50ab2a0945977c335bffc9792300

Scenario 1: Changing the Chain Id in one place

Step 4: Manipulate VAA

0000000000000001 == 1 to 0000000000000002 == 2

Wrong decrypted vaa:-
0100000000010095ee50fbb1c7bd40a5d028664faa9e7dfbaa252e37664375f9926db15fcb13702a
b91acd012fd34bbb8f9d7714ae7cb72b92009e40c23641d70fcbc872aec53f01675959d000000000
000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
002a007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f7347316608
02fc0000000000000002434354505553444300000000000000000000000000000000000000
0000000000000000000000000010000000000000000000000000000000000000000000000000
000000000000c35000522cace9a39b7e9ba32a963739b8776c4327a720f2213de8b7a1fcce6eddd
43dd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 5: Encrypt and Test Altered VAA

Encrypted wrong vaa:-
AQAAAAABAJXuUPuxx71ApdAoZk+qnn37qiUuN2ZDdfmSbbFfyxNwKrkazQEv00u7j513FK58tyuSA
J5AwjZB1w/LyHKuxT8BZ1lZ0AAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAA
AAAAAAACoAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAAACQ
0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAMNQAFIsrOmjm36boyqWNzm4d2xDJ6cg8iE96Leh/M5u3d
Q90Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA

```
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAJXuUPuxx71ApdAoZk+qnn37qiUuN2ZDdfmSbbFfyxNwKrkazQEv00u7j513FK58tyuSAJ5AwjZB1w/
  LyHKuxT8BZ1lZ0AAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACoAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAACQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAFIsrOmjm36
  boyqWNzm4d2xDJ6cg8iE96Leh/M5u3dQ90Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing the withdraw script in solana and getting error

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

> withdraw
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

secp256k1 unavailable, reverting to browser version
withdraw called


  0 passing (0ms)

wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
wallet Uint8Array(64) [
   23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
  225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
  226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
  190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
  247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
   93,  84, 105,  26,  42, 104, 147, 220, 196
]
wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
signedMsg <Buffer 01 00 00 00 00 01 00 95 ee 50 fb b1 c7 bd 40 a5 d0 28 66 4f aa 9e 7d fb aa 25 2e 37 66 43 75 f9 92 6d b1 5f
cb 13 70 2a b9 1a cd 01 2f d3 4b bb 8f 9d ... 259 more bytes>
SendTransactionError: failed to send transaction: Transaction precompile verification failure InvalidAccountIndex
    at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
  logs: undefined
}
```

Scenario 2: Changing the Chain Id in both the place

Step 4: Manipulate VAA

0000000000000001 == 1 to 0000000000000002 == 2

Wrong decrypted vaa:-
010000000001005f7a2695046d14a9a2f25097ca7781048e1386e6b538070e9ff078c42aa3a8d17d
93e174cd8f428b50214b621246c2bf80f9fc7477ecd538399df789316ae2f200675960ad00000000
000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
002b007769746864726177 0f3c69785cf511897365fe0b1c373ffc51bf35076d020ef02f7347316608
02fc000000000000000243435450555344430000000000000000000000000000000000000000
000000000000000000000000000200000000000000000000000000000000000000000000000000
000000000000c35000e99ac30f621138411b4a3ac9feb8c080e646c673abe7d07c1e804392ac175
34bd18de7d3254dc90b870700f658e86afab8f94c50ab2a0945977c335bffc9792300

Step 5: Encrypt and Test Altered VAA

Encrypted wrong vaa:-

AQAAAAABAF96JpUEbRSpovJQl8p3gQSOE4bmtTgHDp/weMQqo6jRfZPhdM2PQotQIUtiEkbCv4D5/HR37NU4OZ33iTFq4vIAZ1lgrQAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACsAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/8Ub81B20CDvAvc0cxZggC/AAAAAAAAACQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAOmaww9iETHBG0o6yf64wIDmRsZzq+fQfB6AQ5KsF1NL0Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA

```
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAF96JpUEbRSpovJQl8p3gQSOE4bmtTgHDp/weMQqo6jRfZPhdM2PQotQIUtiEkbCv4D5/
  HR37NU4OZ33iTFq4vIAZ1lgrQAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACsAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAACQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAOmaww9iETh
  BG0o6yf64wIDmRsZzq+fQfB6AQ5KsF1NL0Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing the withdraw script in solana and getting error

```
gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

> withdraw
> npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

secp256k1 unavailable, reverting to browser version
withdraw called


  0 passing (0ms)

wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
wallet Uint8Array(64) [
   23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
  225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
  226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
  190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
  247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
   93,  84, 105,  26,  42, 104, 147, 220, 196
]
wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
  _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
}
signedMsg <Buffer 01 00 00 00 00 01 00 5f 7a 26 95 04 6d 14 a9 a2 f2 50 97 ca 77 81 04 8e 13 86 e6 b5 38 07 0e 9f f0 78 c4 2a
a3 a8 d1 7d 93 e1 74 cd 8f 42 8b 50 21 4b ... 259 more bytes>
SendTransactionError: failed to send transaction: Transaction precompile verification failure InvalidAccountIndex
    at Connection.sendEncodedTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CO
DEX-dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5923:13)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at Connection.sendRawTransaction (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-
dev/dev/contracts/chains/solana/node_modules/@solana/web3.js/src/connection.ts:5880:20) {
  logs: undefined
}
```

Step 6: Re-test with Correct VAA

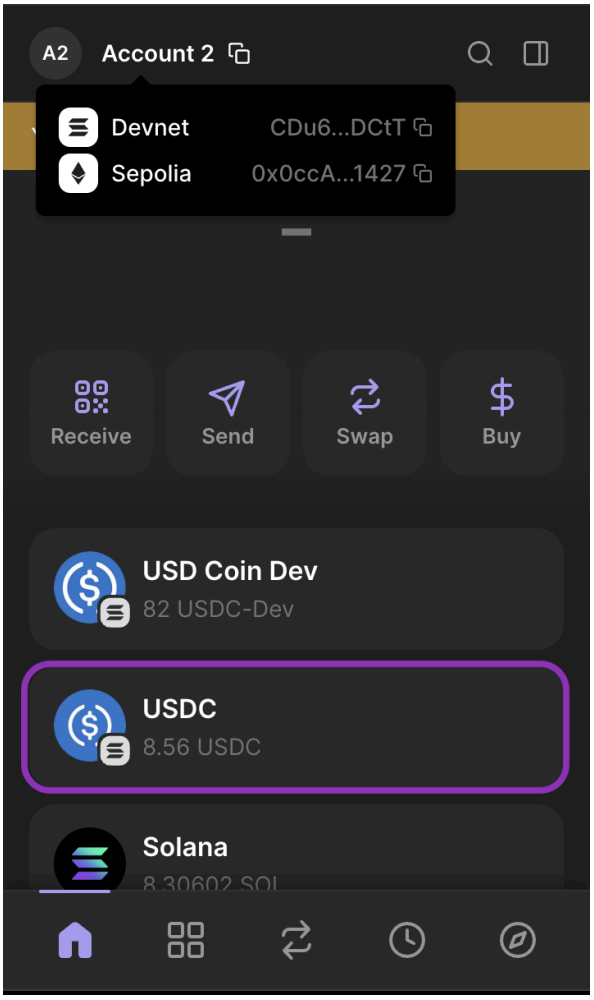
Entering the correct vaa

```
> withdraw                                                    Aa ab .* 4 of 5
async function withdraw() {
  console.log("withdraw called");
  const bufferRaw = "AQAAAAABAJXuUPuxx71ApdAoZk+qnn37qiUuN2ZDdfmSbbFfyxNwKrkazQEv00u7j513FK58tyuSAJ5AwjZB1w/
  LyHKuxT8BZ1lZ0AAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACoAd2l0aGRyYXcPPGl4XPURiXNl/gscNz/
  8Ub81B20CDvAvc0cxZggC/
  AAAAAAAAAABQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAFIsrOmjm36
  boyqWNzm4d2xDJ6cg8iE96Leh/M5u3dQ90Y3n0yVNyQuHBwD2WOhq+rj5TFCrKglFl3wzW//JeSMA";
  const signedMsg = Buffer.from(bufferRaw, 'base64');
  // console.log(signedMsg,"This is the raw vaa")
  // console.log(Buffer.from(signedMsg).toString("hex"),"This is the decoded message from algorand vaa")

  const senderAddress = "22UYZV5RvVhwazuriQNWY2CL44YFEMKDGkX33vWEvj87" //it is destAddress in python script
  const userTokenAccount = new PublicKey(senderAddress);
  const userAccount = await getWalletFromTokenAccount(connection, userTokenAccount);
  await postSignedMsgAsVaaOnSolana(signedMsg);

  await expectIxToSucceed(
    tokenManager.createWithdrawTokenInstruction(
      connection,
      TOKEN_MANAGER_PID,
      payer.publicKey,
      DEVNET_CORE_BRIDGE_PID,
      signedMsg,
      userAccount!,
      userTokenAccount,
      0,
      true
    )
  );
}
```

Executing Withdraw in solana with correct vaa

```
● gurkiratsingh@Gurkirats-MacBook-Air solana % npm run withdraw

  > withdraw
  > npx ts-mocha -p ./tsconfig.json -t 1000000 ts/scripts/main.ts --withdraw

  secp256k1 unavailable, reverting to browser version
  withdraw called


    0 passing (1ms)

  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  wallet Uint8Array(64) [
     23,  47, 111, 213, 195, 235,  16, 180, 130, 243, 195,
    225,  86,  91, 242,  72,  55,  38, 253,  54, 190,  44,
    226,  56, 164, 208,  97, 130, 146,  20, 127,  69, 166,
    190, 112,  64, 123, 111,  94,   3,  62, 103, 117, 239,
    247, 210,  83,  20, 205, 229, 124,  13, 184,  41,  47,
     93,  84, 105,  26,  42, 104, 147, 220, 196
  ]
  wallet PublicKey [PublicKey(CDu6x7LTF6quqrdJHinxMUA6YpGsSuVfYenXvFx6DCtT)] {
    _bn: <BN: a6be70407b6f5e033e6775eff7d25314cde57c0db8292f5d54691a2a6893dcc4>
  }
  signedMsg <Buffer 01 00 00 00 00 01 00 95 ee 50 fb b1 c7 bd 40 a5 d0 28 66 4f aa 9e 7d fb aa 25 2e 37 66 43 75 f9 92 6d b1 5f
  cb 13 70 2a b9 1a cd 01 2f d3 4b bb 8f 9d ... 259 more bytes>
  response [
    {
      signature: '5gQqQLoBbTvzSticiuX1We8hvseNpGRKuv4QHpJbnACtLYjF1E2tWAfMBnAmzYUXMQaX2ByUAdZPZae7GdnToZhE',
      response: { context: [Object], value: [Object] }
    },
    {
      signature: '5CkaEYaRpqZ5W2Ydvt4kygMUevapGQQcfJGMmTVZcKy6wfpZWJgxXAwvUJqSKpACgjnrroENuMwTKyQaicKtGdNo',
      response: { context: [Object], value: [Object] }
    }
  ]
  createWithdrawTokenInstruction is called
  token PublicKey [PublicKey(4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU)] {
    _bn: <BN: 3b442cb3912157f13a933d0134282d032b5ffecd01a2dbf1b7790608df002ea7>
  }
  receivedAcc PublicKey [PublicKey(BwJdo3WBw2BSUST2jpUkrQxfQhAUJhMMEAMMJdV4hPey)] {
    _bn: <BN: a27e2c15bfa7b0ac0ed5a23af3e59431fdb10c26bda40db3d80859e155940022>
  }
  posted PublicKey [PublicKey(GReyh7nHdfKpAVQGUJhvZZNFhs9w5Ui8hCtgNjhusZUh)] {
    _bn: <BN: e5302f3e4c2a278da470ef834e8381bdee2cedd4a8c6656f2f21e79801d442d6>
  }
```

Balance updated of the user EOA gets updates with correct vaa



Transaction id on Solana is :-
3×9AGanhb74Kmw2VDCGxRXRXC3HCkSJNCaowRARfD69QTpPg9JcFSKwHeHohkGVubKwyfw2PxmLuPJVDMFkMjFgB

5. Auditing Approach and Methodologies Applied

The Deposit and withdrawal transactions were tested in a comprehensive approach to ensure the highest level of security and reliability. Careful attention was given to the following key areas to ensure the overall quality of code:

- **Code quality and structure**: We conducted a detailed review of the codebase to identify any potential issues related to code structure, readability, and maintainability. This included analyzing the overall architecture of the smart contract and reviewing the code to ensure it follows best practices and coding standards.
- **Security vulnerabilities**: Our team used manual techniques to identify any potential security vulnerabilities that could be exploited by attackers. This involved a thorough analysis of the code to identify any potential weaknesses, such as buffer overflows, injection vulnerabilities, Signatures, and deprecated functions.
- **Documentation and comments**: Our team reviewed the code documentation and comments to ensure they accurately describe the code's intended behavior and logic. This helps developers to better understand the codebase and make modifications without introducing new issues.
- **Compliance with best practices**: We checked that the code follows best practices and coding standards that are recommended by the community and industry experts. This ensures that the smart contract is secure, reliable, and efficient.

Our audit team followed OWASP and Solana community security guidelines for this audit. As a result, we were able to identify potential issues and provide recommendations to improve the smart contract security and performance.

Throughout the audit of the smart contract, our team placed great emphasis on ensuring the overall quality of the code and the use of industry best practices. We meticulously reviewed the codebase to ensure that it was thoroughly documented and that all comments and logic aligned with the intended behavior. Our approach to the audit was comprehensive, methodical, and aimed at ensuring that the smart contract was secure, reliable, and optimized for performance.

6. Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of the Ultrade Solana Project and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Ultrade Solana smart contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report based on changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by Entersoft for the exclusive benefit of Ultrade Solana and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Ultrade Solana governs the disclosure of this report to all other parties including product vendors and suppliers.