Bridging Ideation to Feature Completion

What are we going to be covering?

- This workshop will guide you through the comprehensive process of turning an idea into a fully realized feature, using the example of ***cross-chain bridging*.**

- The session will cover requirement analysis, solution proposal, basic threat modeling, edge case identification, solution revision, and testing. We will also explore the importance of defining test cases, addressing edge cases during development.

What are we solving?

Requirement:

Develop a cross-chain bridge between Binance Smart Chain (BSC) and Ethereum to enable the seamless transfer of a custom ERC-20 token with a limited supply of 18,000,000,000 tokens.

The primary focus is on ensuring security, especially preventing double-spending, and maintaining a decentralized system as much as possible.

1. Should be secured.
2. No double spending attacks to be relayed.
3. Emergency exit planning
4. Platform will not shoulder the cost of transaction fees required on both networks

Standard Software Functioning Practices

1. Disable certain functionality triggers in case the user doesn't have what it requires to use the functionality.
2. All the methods should have input validation enabled along with business logic incorporated.
3. Error handling for smooth user flow integration and avoid leaving the user dumbfounded.
4. **Threat model the solution** which requires coming up with scenarios that will break the solution/logic in place.
5. **Test the solution using different types of testing [unit, performance, integration, security, regression, UI]**

Proposed Solution

1. Lock and Mint, Burn and Unlock Model will be used.
2. User will pay their claim cost on the secondary chain
3. Multisig will be used for signing the claimable transaction
4. Relayer will be used which would have access to 5 validators
5. The threshold would be 3.
6. Event listener to alert the user over the transaction completion.
7. Have a contract counter and a destination and source chain nounce for message hash.
8. Have a DB in which user transaction details will be stored – [fromHash, amount, CreatedAt, fromChain, isMigrated, messageHash, receiver, signatures[], toChain, updatedAt, toHash]

User Flow

Blog link for walkthrough demo: LINK

Token Approval:

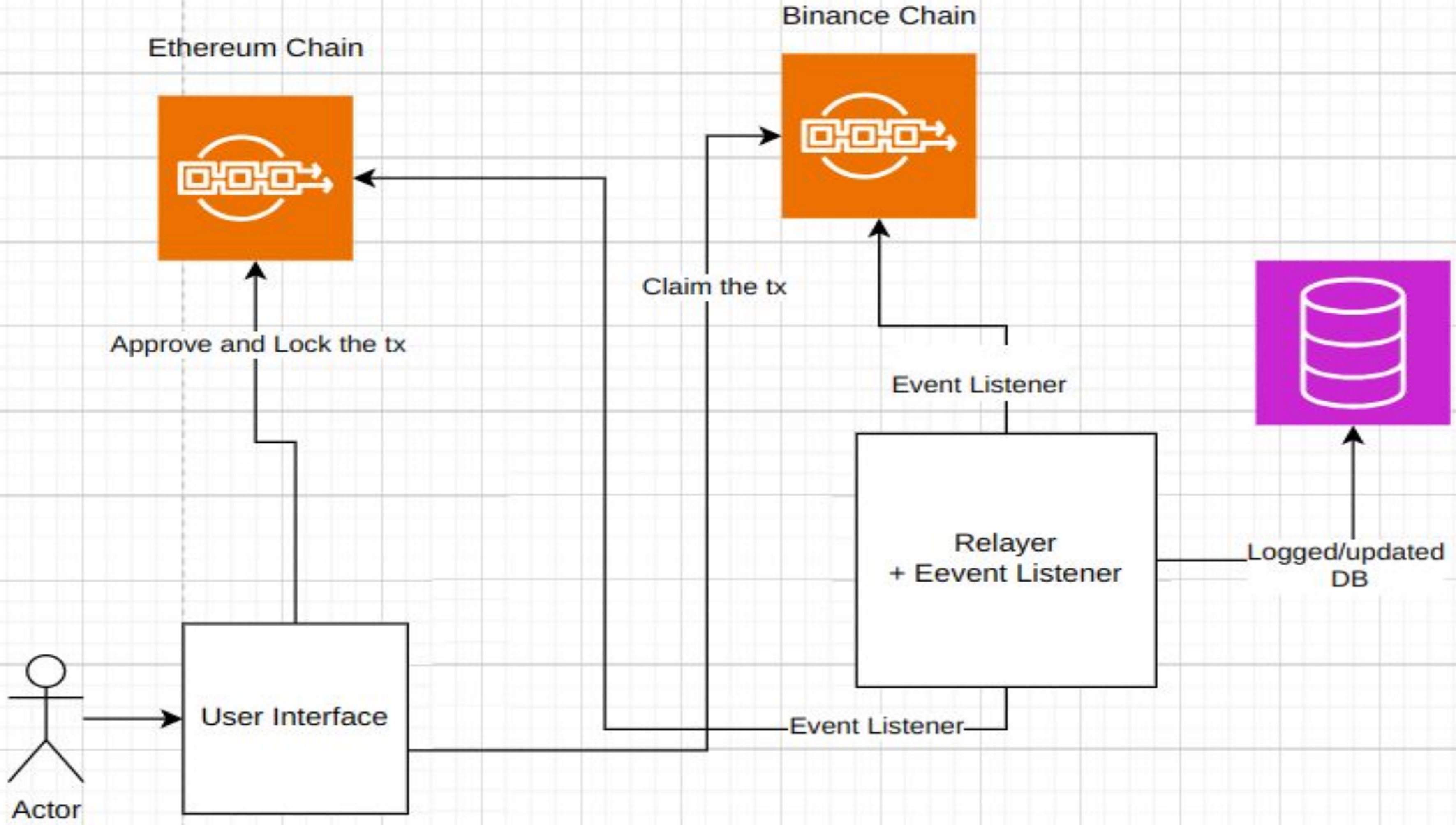
- The user approves the custom ERC-20 token to be used by the cross-chain bridge contract on the Ethereum network.

Transaction Execution:

- The user ensures they have sufficient balance for the transaction fees.
- The user initiates the transaction by signing it and executing it on the Ethereum network.

User Claim:

- The user pays the claim cost on the Binance Smart Chain.
- The user claims the tokens on the Binance Smart Chain.


Ethereum Chain
Binance Chain
Claim the tx
Approve and Lock the tx
Event Listener
Relayer
+ Eevent Listener
Logged/updated
DB
User Interface
Event Listener
Actor

What Are The Attack Vectors? Threat Modeling

1. Where are we storing the keys? Private keys through which the user transaction is signed?
2. Directly interact with the bridge contract? It wont work because of relayer dependency for signatures.
3. The approval amount to the contract? Incase contract acts maliciously?
4. Try to claim the same tx but with different chain.
5. Multiple validators collude to manipulate the transaction validation process.
6. Compromise of the private keys used by validators or the relayer.
7. Relayer Get Request exploit (not significant)

Possible checks/test cases for smooth user flow

1. Sufficient balance check.
2. Input field checks
3. Information icons
4. Native currency check in the account balance for transaction initiation.
5. Incase Metamask throws an error?
6. Contract exception error handling
7. Address validity verification before initiating tx
8. Address uppercase and lowercase check.
9. Processing Modal until the transaction is sent.
10. Display transaction hash and navigate the user to relevant explorer.
11. Claim status update incase the user comes up later, isMigrated should be updated timely to show user recent claim tx statuses.
12. Include the processor until the the claim is added in the DB by relayer.

Attack Vectors and Threat Modeling when Platform pays the fees

1. Calculation of gas fees incase of network traffic.
2. How to ensure the wallet is funded with native currency to pay for the user transaction fees readily.
3. What if the wallet is drained of native currency how will the admin be informed to fund the wallet timely.
4. Slippage calculation as the fees from the user will be deducted in tokens.

User Flow testcases when platform pays the fees?

The fees will be deducted from the user entered token amount he has instead of the native currency.

1. The user should be shown the complete breakdown of the amount he is migrating.
2. The user should be able to calculate the required fees in tokens regardless he has sufficient balance or not
3. The user should be should be guided over the time it will take for the migration to go through.
4. The user should be notified once the migration is completed.
5. The user should be informed about the status of the migration as well.
6. The user should be able to see its past migrations.
7. Because the user will pay the fees in migration tokens then the quote which will be calcuated should have an expiry as the tokens are volatile and its not a stable token.

Pour in Feedback