Blockchain Phase 2 Report

Problem Statement -

Decentralized finance (DeFi) has revolutionized the financial industry by providing borderless, permissionless, and transparent access to financial services. However, the rising popularity of DeFi has led to a surge in security risks, including sandwich attacks on Uniswap v3 transactions, where malicious actors manipulate the market by front-running transactions and causing price slippage. This problem poses a significant threat to the DeFi ecosystem, as users can potentially lose a substantial amount of funds in a single transaction. Therefore, there is an urgent need to develop robust security solutions that can mitigate these risks and ensure the long-term sustainability of DeFi.

BackGround

- **DEFI:** DeFi is a new form of finance that utilizes decentralized blockchain technology to provide financial services. It eliminates intermediaries, such as banks, and allows users to have full control over their funds.
- **Liquidity Pools:** Liquidity pools are a crucial component of decentralized exchanges such as Uniswap. They are pools of tokens that are used to facilitate trades, and they allow for greater liquidity and price discovery. The more liquidity a pool has, the easier it is to buy and sell tokens on the exchange.
- **Automated Market Makers (AMM):** AMMs are algorithms that allow for the automatic trading of digital assets by using a liquidity pool. The pool is created by users who deposit their tokens into it, and the algorithm sets the price based on supply and demand.
- **UNISWAP:** Uniswap is a popular AMM platform that allows users to swap tokens without needing an order book or counterparty. It uses a constant product market maker model that ensures liquidity by adjusting the price based on supply and demand.
- **Smart Contract Auditing:** Smart contract auditing is the process of reviewing and analyzing smart contracts for security vulnerabilities. Given the large amount of money that can be at stake in DeFi projects, it is crucial to ensure that smart contracts are thoroughly audited to prevent hacks and other security issues.
- **Token Swap:** A token swap is a transaction where one cryptocurrency is exchanged for another. It can be done manually or automatically through a platform like Uniswap.
- **Price Impact:** Price impact is the change in price caused by a transaction in a liquidity pool. It is determined by the amount of tokens being traded and the liquidity available in the pool.
- **Mempool:** The mempool is a data structure that holds all pending transactions on the blockchain network. It is where the transactions wait to be picked up by a miner and added to the blockchain.

- **Order Reordering:** Order reordering is a technique used by miners to maximize their MEV by changing the order of transactions in the mempool to their advantage. It can cause front-running and other market manipulation tactics.

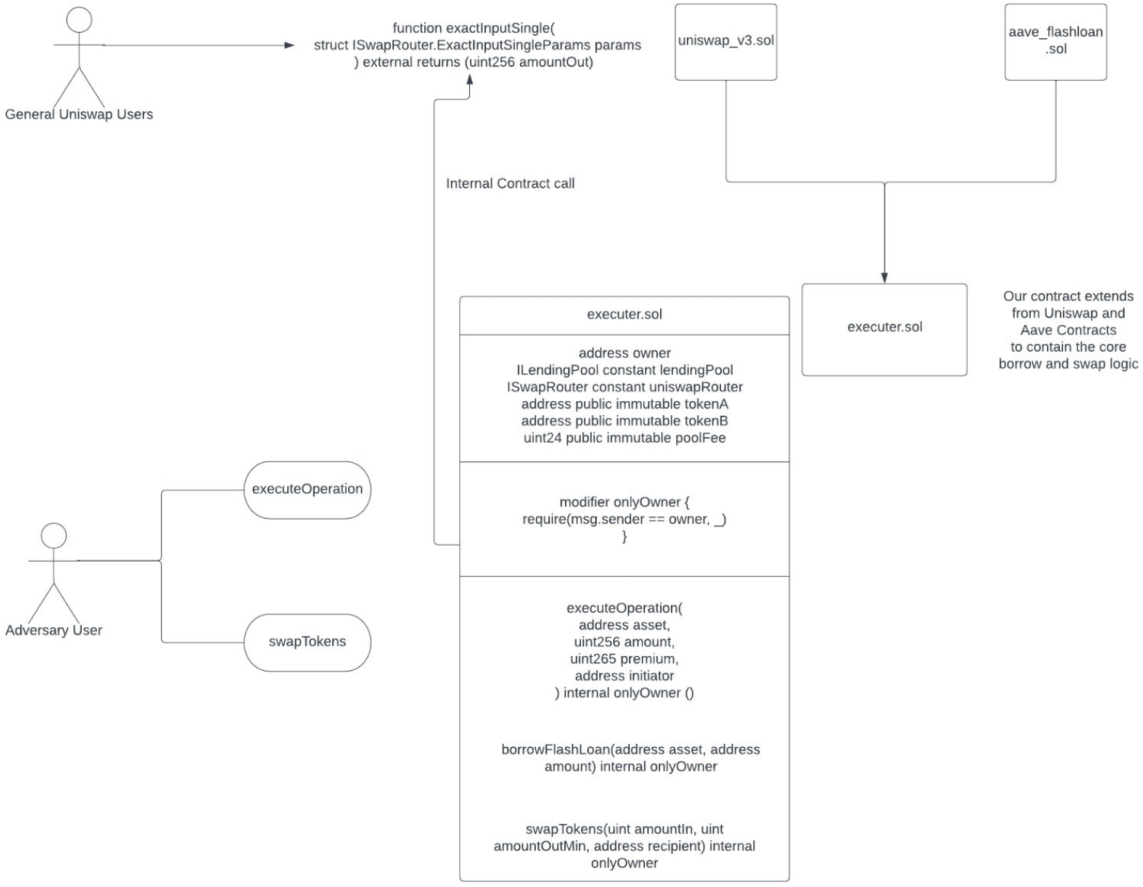
METHODOLOGY

1. **Monitoring the Mempool:** The first step is to monitor the Ethereum Mempool for any pending Uniswap transactions. This involves using web3.js and the Ethereum network architecture to keep track of all incoming transactions.
2. **Decoding Transaction Data:** Once a transaction is detected, the bot will decode the transaction data using the Uniswap v3 decode function to extract information on the token swaps, amounts, gas fees, etc.
3. **Calculating Expected Output and Slippage:** The bot will then determine the expected output of the transaction based on the current price of the tokens in the Uniswap pool. It will then calculate the slippage, or the amount of price impact the transaction can tolerate, which will determine the maximum extractable value.
4. **Taking a Flashloan and Calculating Variables:** After decoding the transaction data, the bot will take a flashloan and calculate all the necessary variables for the flashloan, including the amount of assets to borrow, the current market price, the Uniswap fee, and the gas fee for both transactions. It will ensure that the extracted value from the sandwich attack is higher than all of these expenses.
5. **Placing Transactions in the Uniswap Pool:** If the transaction is profitable, the bot will execute the sandwich attack by placing buy and sell orders around the victim's transaction, effectively sandwiching it. The bot will then monitor the Uniswap pool dynamics to determine the new sqrt96ratio and tick. It will calculate the new price and compare it to the original price to determine the price impact caused by the transaction.
6. **Avoiding Failed Transactions:** The bot will ensure that the price impact is not greater than the slippage to avoid a failed transaction. If the price impact is within the tolerable slippage range, the bot will extract maximum value from the transaction.
7. **Continuously Monitoring the Pool:** The bot will continuously monitor the Uniswap pool dynamics to ensure that the transaction remains profitable. If the pool dynamics change and the transaction is no longer profitable, the bot will cancel the transaction to avoid any losses.
8. **Implementing Advanced Strategies:** To further increase profits, the bot can implement advanced strategies such as order reordering and MEV extraction. Order reordering involves analyzing the Uniswap order book to place orders in a way that maximizes profits. MEV extraction involves analyzing the Mempool for profitable transactions that include miner extractable value (MEV) and taking advantage of them.
9. **Ensuring Security:** As the bot will be dealing with sensitive financial information and transactions, ensuring security is crucial. This involves implementing secure smart contract code using Solidity programming and following best practices for secure software development.
10. **Displaying Results:** Once the bot has executed the sandwich attack or determined that the user's transaction is safe, it will display a breakdown of all the transactions involved in the attack and what would happen if the user placed their transaction on the application UI.

11. **Directly Signing Transactions:** If the user wants to place a transaction, they can directly sign the same transaction from our webpage rather than going to Uniswap and placing it again. This saves the user time and ensures that the transaction is executed correctly.

Contract Diagram

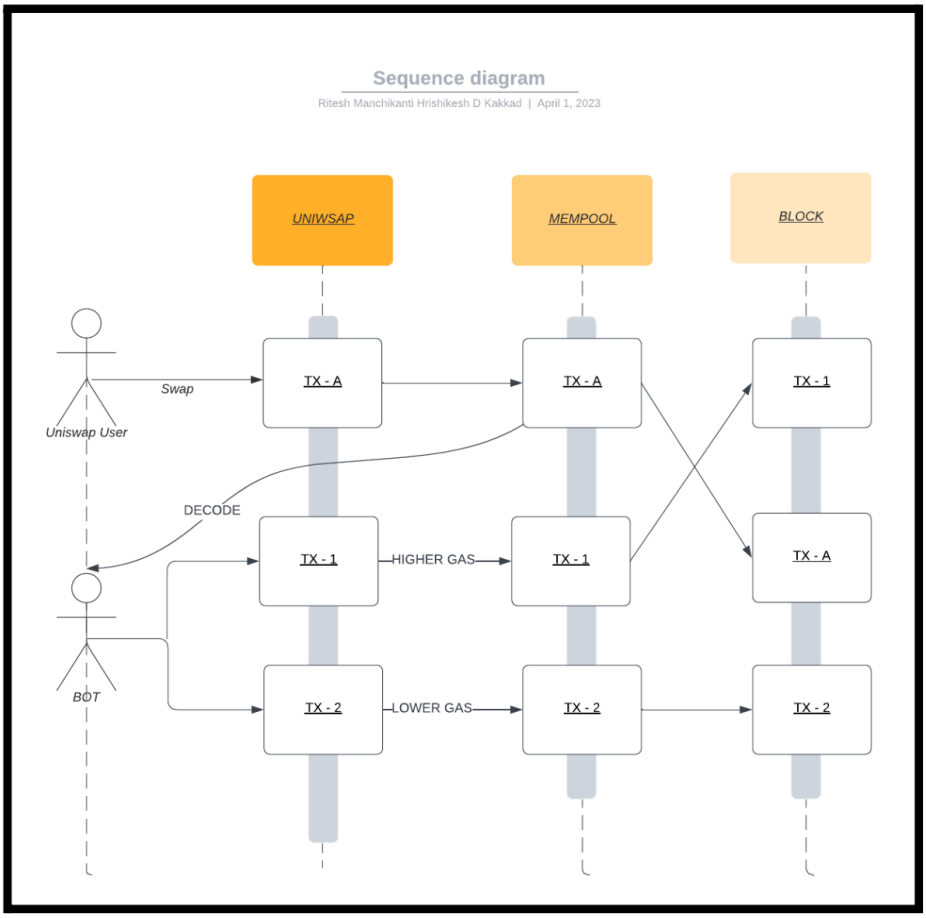
Use Case and Contract Diagram



-

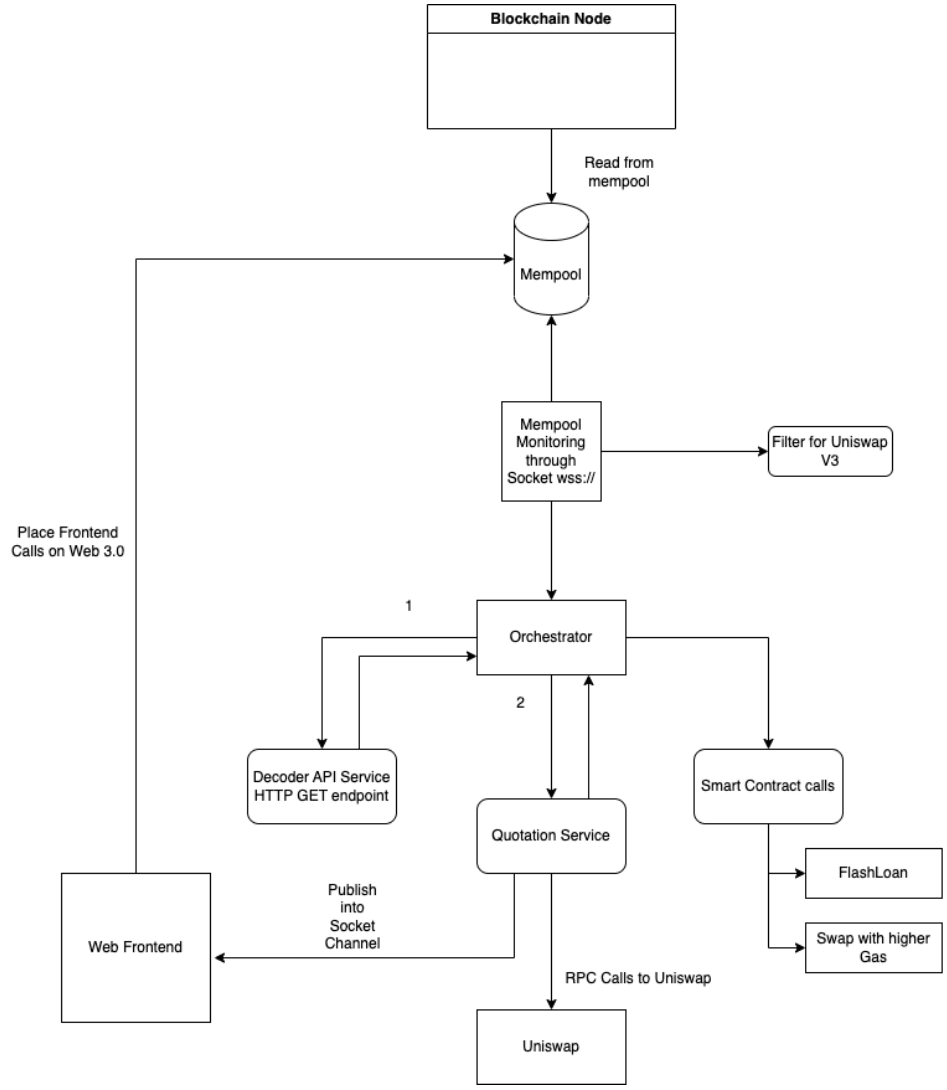
Sequence Diagram

Sequence Diagram



-

Architectural Diagram



-

Steps for Deployment

- Hit npm install inside the following folders:
  - Root
  - Universal-decoder
  - Frontend-simulator
- Hit npm start for all the folders in a new tab
- Create a .env file and add your alchemy endpoint as defined in the .env.example file
- From the root folder - Run $npx hardhat scripts/deploy.ts. This will deploy all the relevant contracts to Goerli.
- Start your local Hardhat node as well
- Start the mempool watcher script by hitting the command - **ts-node scripts/mempool.ts**
- Now set your meta mask wallet on HardHat
- Go to the UniSwap interface and place and sign a swap
- You will see all the information about the transaction on the frontend in real-time
- Your **SETUP** is now complete

Conclusion

In conclusion, this project aimed to develop a bot that can monitor Uniswap transactions and execute a sandwich attack to extract maximum value from these transactions. The methodology involved monitoring the Ethereum Mempool, decoding transaction data, calculating expected output and slippage, placing transactions in the Uniswap pool, and implementing advanced strategies such as order reordering and MEV extraction.

To further increase profits, the bot used flashloans to borrow a large amount of assets without collateral and calculated all the necessary variables to ensure that the extracted value from the sandwich attack was higher than all the expenses. The bot also ensured security by implementing secure smart contract code using Solidity programming and following best practices for secure software development.

The results of the bot were displayed to the user through a UI, which indicated whether their transaction was safe or vulnerable and provided a breakdown of all transactions involved in the sandwich attack. If the user decided to place a transaction, they could directly sign the same transaction from the webpage.

Overall, this project demonstrated the potential of algorithmic trading and financial analysis in the world of decentralized finance. The development of this bot can help users extract maximum value from their Uniswap transactions while also ensuring security and minimizing risks.