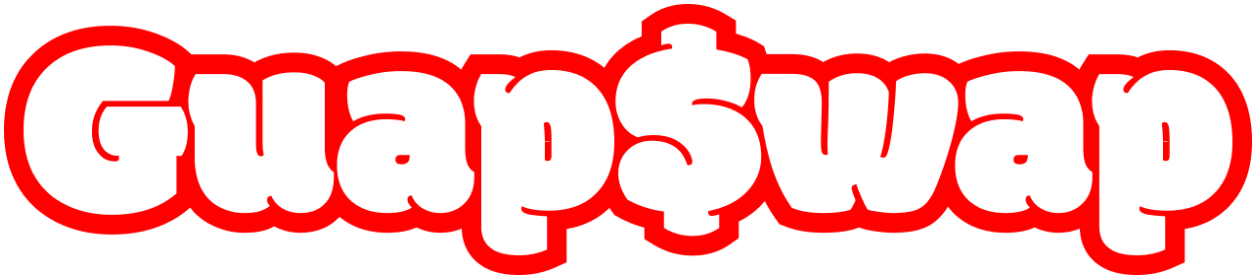


ErgoHack 3 Project Report

GuapSwap Team

Description

This is a report documenting our progress for the GuapSwap project during ErgoHack 3. We describe the relevant sections of the code base that we have implemented thus far. We include the UI mockup, a Todo list of items we have not completed but will in the future, and a product comparison showcasing their pros and cons with respect to GuapSwap.

Links

GitHub: https://github.com/GuapSwap/guapswap/tree/v0.1.0
Twitter: https://twitter.com/GuapSwapErgo
Email: guapswap@protonmail.com

Documentation

User Settings

In order to determine the node settings and the parameter settings that are specific to the GuapSwap protocol and ErgoDex, the user will need to modify a file called guapswap_config.json shown below. We use these variables throughout the program, which are stored in an object called GuapSwapConfig

```
{
    "node": {
        "nodeApi": {
            "apiUrl": "",
            "apiKey": ""
        },
        "wallet": {
            "mnemonic": "",
            "password": "",
            "mnemonicPassword": ""
        },
        "networkType": "TESTNET"
    },
    "parameters": {
        "guapswapProtocolSettings": {
            "userAddress": "",
            "serviceFees": {
                "protocolFeePercentage": 0.0025,
                "protocolUIFeePercentage": 0.0,
                "protocolMinerFee": 0.002
            }
        },
        "dexSettings": {
            "ergodexSettings": {
                "swapAssetTicker": "SigUSD",
                "slippageTolerancePercentage": 0.001,
                "nitro": 1.2,
                "ergodexMinerFee": 0.002
            }
        }
    }
}
```

Commands

The commands used to interact with the program include the following:

$ guapswap generate
- generate proxy address from guapswap_setting.json

$ guapswap swap <proxy_address>
- loops indefinitely and queries blockchain once every 60 min to check for a payout and performs the swap automatically.

$ guapswap swap <proxy_address> [–onetime]
- perform a one time swap with all utxo at this address

$ guapswap refund <proxy_address>
- refund all utxo at this address to miner

$ guapswap list <proxy_address>
- list all utxo with given proxy address

$ guapswap log
- show error logs of program

$ guapswap exit
- terminate program

Unfortunately, we ran out of time and were not able to implement these functions. However, in order to help with creating the Scala CLI in the future, we found a third-party library called *decline*: https://github.com/bkirwi/decline

Contracts

GuapSwapErgoDexSwapSellProxyContract

This is the proxy contract that is compiled and generated for the user, which they subsequently provide to a mining pool of their choice. Their payouts will be sent to this address. The contract script can be seen in our GitHub repository here:

https://github.com/GuapSwap/guapswap/blob/v0.1.0/src/main/scala/contracts/GuapSwapErgoDexSwapSellProxyContract.scala

GuapSwapServiceFeeContract

This service fee charged by the GuapSwap protocol will be sent to this contract. In order to spend the funds, a threshold amount will need to be met and locked funds will be split amongst the team members. Unfortunately, we have not implemented this contract yet.

ErgoDexSwapSellParams

These are the parameters necessary to perform the swap with ErgoDex. These are provided to the proxy contract as Context Variables. The proxy contract uses these variables to substitute the constants in the ErgoDex swap-sell contract, whose proposition bytes are provided to the proxy as a hard-coded constant. The substitution occuring within the proxy contract is compared to the substitution done outside the contract, and the two scripts are compared as a spending condition for a valid ErgoDex swap box output.

Sell Ergs

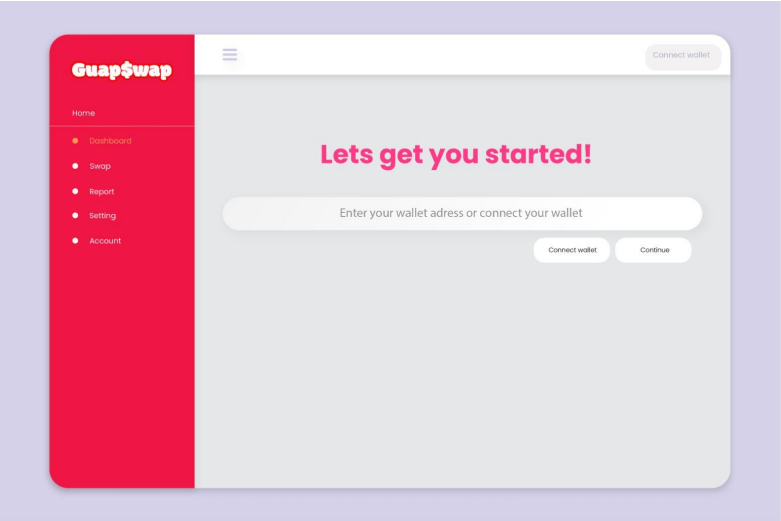
Contract parameters

Constant	Type	Description
Pk	SigmaProp	User PublicKey
FeeNum	Long	Pool fee numerator (must be taken from pool params)
QuoteId	Coll[Byte]	Quote asset ID. This is the asset we are buying from the pool
MinQuoteAmount	Long	Minimal amount of quote asset
BaseAmount	Long	The amount of nanoErgs to sell
DexFeePerTokenNum	Long	Numerator of the DEX fee in nanoERGs per one unit of quote asset
DexFeePerTokenDenom	Long	Denominator of the DEX fee in nanoERGs per one unit of quote asset
MaxMinerFee	Long	Max miner fee allowed at execution stage
PoolNFT	Coll[Byte]	ID of the pool NFT (Used as a reference to a concrete unique pool)
MinerPropBytes	Coll[Byte]	Miner script

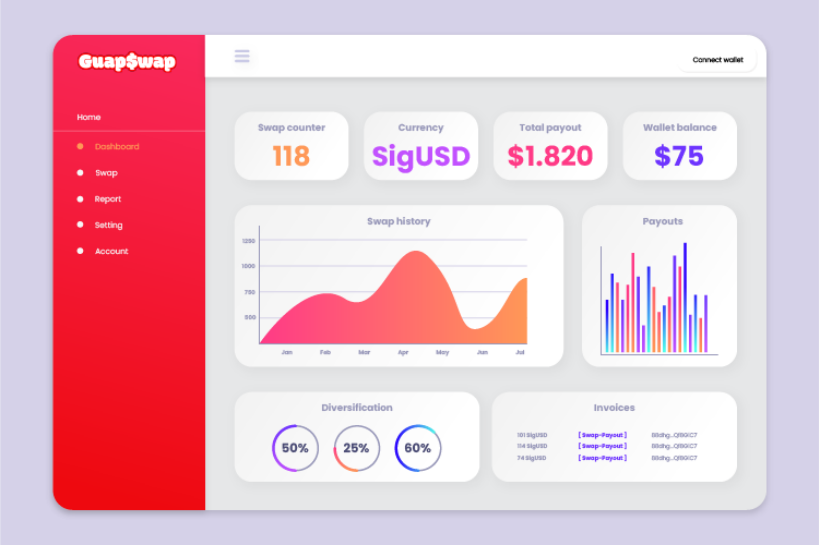
UI Layout Mockup

Steps

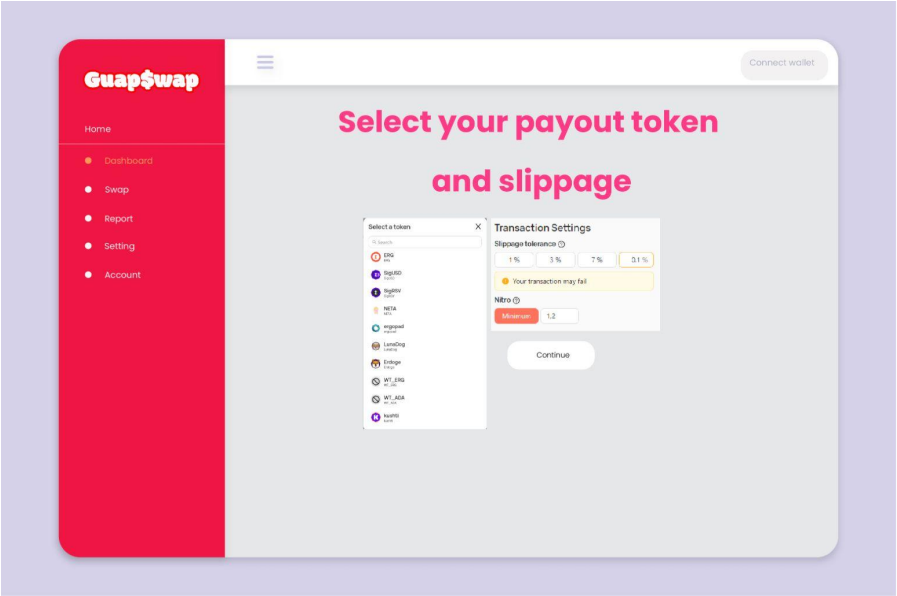
1. User inputs the public key manually or connects the wallet.



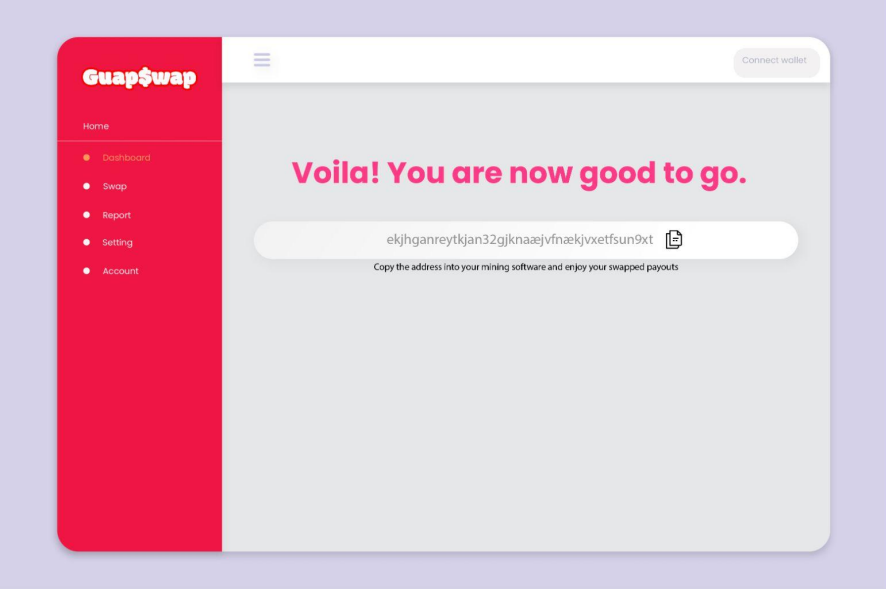
2. User is presented with a dashboard showing following information:
    - “Create New Address” button
    - Already generated addresses(if any)
    - Slippage rates for already generated addresses
    - Total Ergo paid to each address
    - Total tokens paid out to wallet from each address



3. User selects the “Create New Address” option.
4. User selects the payout token they want and the slippage tolerance.



5. The user is presented with the proxy contract address which they can paste into their mining software.



6. User is redirected back to the dashboard now showing the newly generated address.

Todo

- Implement commands
- Write Service Fee contract
- ErgoSript contract review/audit
- Running a v0.1.0 on testnet or mainnet, whichever has the execution bots running.

Product Comparison

When looking at comparable products we have to keep in mind that GuapSwap is the first product to offer any mining payout swapping service on the Ergo Blockchain. With that in mind we can look at three other products that have similar functions and provide swapping for mining payouts.

GuapSwap

GuapSwap is a fully decentralized smart contract profit swapping service on the Ergo Blockchain. To use GwapSwap the client either interacts with GwapSwaps UI to generate a proxy contract address for payouts or they run the CLI on a personal node. This allows any mining pool to be used for payouts. After a pool pays out to the proxy contract address it will be automatically swapped for the token the client chose through ErgoDex and deposited in the client's wallet.

Pros:

- Fully decentralized, does not interact with any CEX.
- Much faster clearance of swaps by using a dex.
- Reduces taxable events for clients.
- Fully transparent swapping method.
- First smart contract miner profit swapping expediter.
- Interacts with any Ergo Pool allowing clients to use their favorite pool and interface to track mining rewards.

Cons:

- Adds another step to the mining process by interacting with the proxy contract.
- Additional fees

UnMinable (Multicoin)

UnMinable is a fully centralized mining service, they even offer their own mining software to use with their pool. Their swapping service works like this, a miner will download UnMinables .exe and run it, picking a payout coin or token from their provided list of options. After choosing a coin they enter their wallet address they choose from the available algorithms they can mine and hit the start button. When the miner reaches their payout threshold they can request a payout on UnMinables website. All this is made possible by unmineable swapping coins on a CEX before sending the payout.

Pros:

- Easy to understand and use for beginners, streamlined interface.
- Reduces taxable events for clients.
- Lets of different tokens on many different chains.
- Able to mine with different algorithms (ETH, ETC, KApow, RandomX)

Cons:

- Fully centralized using CEX for payouts.
- Lack of transparency is regards to payouts (usually payouts are notably lower that estimated earnings of pure ETH)
- Lack of transparency in regards to CEX used for swaps.
- Unable to see pool stats in the interface.
- Proprietary Mining software is inefficient not providing full hashrate.
- Only able to use one mining pool for service.

2 Miners (BTC or NANO payout)

2 miners is an ETH mining pool that allows for payouts in either BTC or NANO. The miner sets up their mining software to mine to the 2 Miners pool and enters either a BTC, NANO, or ETH address to be paid to them when they reach their payout threshold. 2 Miners then takes this payout grouped with others once a day and sends it to a CEX and swaps it for the correct Coin and then sends it to the miners address.

Pros:

- Alleviates high gas fees from receiving an ETH payout.
- Reduces taxable events for clients.
- No transaction fees for Nano payouts.

Cons:

- Fully centralized using CEX for payouts.
- While more transparent about swaps then UnMinable they still do not tell you what CEX service they use for swaps.
- Only able to use one mining pool for service.

NiceHash (BTC Payouts Only)

NiceHash is a fully centralized hashrate rental service that offers mining of many coins including ERGO but they only offer payouts in BTC and no other coin. To use their service someone would download their software and register on their site. Run the software and let it calculate what the most profitable coin is to mine before starting.

Pros:

- Reduces taxable events for clients (assuming they want a BTC payout in the end).
- Easy to understand and use for beginners, streamlined interface.

Cons:

- People can rent hashrate from NiceHash opening coins up to 51% attacks if someone has the equity for a large enough rental.
- Lack of transparency in regards to CEX used for swaps.
- Spotty history of hacks.
- Frozen payouts in times of high sell pressure.
- Doubtful marketing slandering other mining software (Phoenix miner especially).

Protocol Spec

Protocol specification for dApp that enables Ergo miners to swap their miner profit payouts.
Follows EIP-6: https://github.com/ergoplatform/eips/blob/master/eip-0006.md

Stages

- GuapSwap Proxy Contract

Actions

- Generate ErgoDex Swap Box
- Refund

Stage: GuapSwap Proxy Contract

In this stage, the miner has already given their proxy contract address to their mining pool. They have received a payout to a box protected by the proxy contract.

Registers

- None

Hard-coded Values

- PK
- ErgoDexSwapSellContractSample
- GuapSwapServiceFeePercentageNum
- GuapSwapServiceFeePercentageDenom
- GuapSwapServiceFeeContract
- GuapSwapMinerFee
- MinErgoDexExecutionFee

Context Extension Values

- FeeNum
- QuoteId
- MinQuoteAmount
- BaseAmount
- DexFeePerTokenNum
- DexFeePerTokenDenom
- MaxMinerFee
- PoolNFT

Mandatory Stage Spending Conditions

- A valid swap box => check that the output swap box PropBytes match the ErgoDex SampleContractPropBytes but with the updated constant values
- A valid ServiceFeeBox with the appropriate service fee
- Transaction signed by PK => implies a refund initiated

Actions/Spending Paths

- Generate ErgoDex Swap Box
- Refund

Action: Generate ErgoDex Swap Box

For this action, the miner chooses to follow through with swapping their payouts. A valid ErgoDex swap box is created.

Data-Inputs

- PoolBox

Inputs

- ProxyContractBox

Outputs

- ErgoDexSwapBox
- MinerFeeBox
- ServiceFeeBox

Action Conditions

- A valid swap box => check that the output swap box PropBytes match the ErgoDex SampleContractPropBytes but with the updated constant values
- A valid ServiceFeeBox with the appropriate service fee
- Transaction signed by PK => implies a refund initiated

Action: Refund

Instead of going through with swapping the payout profits, the miner chooses to obtain the payout in ERG to their wallet, or spend the box via any other transaction.

Data-Inputs

- None

Inputs

- Proxy Contract Box

Outputs

- If refund initiated via application, then a refund box to the user's wallet. Otherwise, the user can spend the Proxy Contract Box in whatever transaction they want.

Action Conditions

- Transaction signed with PK

Protocol Diagrams