Formula Inu

Smart Contract Audit Report

AUDIT SUMMARY

Formula Inu is a new ERC-20 token on Ethereum that is an automatic liquidity providing protocol.

For this audit, we reviewed the project team's FormulaInu contract at 0x2c022e58c5e3ee213f06f975fd8a0d3a6fe9ca1c on the Ethereum Mainnet.

AUDIT FINDINGS

Informational findings were identified and the team may want to review them. In addition, centralized aspects are present.
Date: December 2nd, 2022.

Finding #1 - FormulaInu - Informational

Description: *Although the withdrawToken() function disallows the owner from withdrawing the $FINU token from the contract, the withdraw() function allows the owner to withdraw the $FINU token from the contract.*

```
function withdraw() external onlyOwner {
    uint256 balance = IERC20(address(this)).balanceOf(address(this));
    IERC20(address(this)).transfer(msg.sender, balance);
    payable(msg.sender).transfer(address(this).balance);
}

function withdrawToken(address _token, address _to) external onlyOwner {
    require(_token != address(0), "_token address cannot be 0");
    require(_token != address(this), "Can't withdraw native tokens");
    uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
    IERC20(_token).transfer(_to, _contractBalance);
}
```

Recommendation: *The team should either remove the* `_token != address(this)` *check from the withdrawToken() function or modify the withdraw() function to only transfer the contract's ETH balance to the owner.*

Finding #2 - FormulaInu - Informational

Description: *The* `manualBurnFrequency` *state variable can never be modified, but is not declared constant.*
Recommendation: *This state variable could be declared constant for additional gas savings on each reference.*

Finding #3 - FormulaInu - Informational

Description: *Although the SafeMath library is utilized, the contract is deployed with Solidity v0.8.9 which has built-in overflow checks.*
Recommendation: *SafeMath could be safely removed to reduce contract size, deployment costs, and gas costs on all transactions that utilize it.*

CONTRACT OVERVIEW

- *The total supply of the token is set to 1 billion $FINU [1,000,000,000].*
- *No mint or burn functions exist, though the circulating supply can be decreased by sending tokens to the 0x..dead address.*
- *At the time of writing this report, there are 399 total token holders. The token allocation is as follows:*
  - *10% of the total supply is stored in a Unicrypt token locking contract and will unlock on January 2nd, 2023.*
  - *The next five EOAs own a cumulative 9.97% of the total supply.*
  - *7.82% of the total supply is in Uniswap liquidity.*
  - *Of that liquidity, 86.25% of the LP tokens are stored in a Unicrypt token locking contract and will unlock on November 21st, 2032.*
  - *12% of the LP tokens belong to the owner.*

- *Blacklisted accounts are prohibited from participating in transfers.*
- *The contract enforces a transfer delay which prevents a transfer from occurring if the user is attempting to buy from Uniswap more than one time per two blocks.*
- *The contract enforces a maximum wallet amount which prevents a transaction from occurring if the recipient's balance will exceed the limit number of tokens (determined by the owner) after the transaction takes place.*
- *The contract enforces a maximum transaction amount which imposes a limit to the number of tokens that can be transferred during any given transaction via Uniswap.*
- *There is a Liquidity fee, Marketing fee, and Dev fee on all transfers via Uniswap where neither the sender nor the recipient is excluded from fees. A separate fee structure can be set by the team to apply different fee percentages depending on whether the user is buying or selling during the transfer.*
- *The tokens collected through fees are stored in the contract address. The tokens are swapped for ETH for the purpose of funding Uniswap liquidity when the following conditions are met:*
  - *The automatic liquidity add functionality is enabled by the team.*
  - *The threshold number of tokens (determined by the owner) in the contract address has been reached.*
  - *The contract is not currently performing an automatic liquidity add.*
  - *The caller is not initiating a buy transaction via Uniswap.*
  - *Neither the sender nor the recipient is excluded from fees.*
- *Liquidity-adds are automatically performed by selling the tokens collected as fees, pairing the received ETH with the token, and adding it as liquidity to the ETH pair.*
- *The LP tokens received through this process are sent to the owner. We recommend the team lock these newly acquired LP tokens.*
- *The tokens collected through the Dev fee are swapped for ETH and sent to the team's Dev wallet.*
- *The remaining ETH in the contract is sent to the team's Marketing wallet.*
- *A percentage of tokens (determined by the owner) in the Uniswap Pair address will automatically be sent to the 0x..dead address on all sells via Uniswap as long as the wait time has passed since this functionality has previously occurred.*
- *As the contract is deployed with Solidity v0.8.9, it is protected from overflows/underflows.*
- *The contract complies with the ERC-20 token standard.*

Ownership Controls:

- *The owner can modify each fee to any percentages as long as the total fee percentage combined does not exceed 10% for both the buy and sell fee structures.*
- *The owner can exclude and include accounts from transfer fees.*
- *The owner can add/remove accounts from the transfer blacklist at any time.*
- *The owner can update the maximum wallet amount to any value greater than 5 million tokens.*
- *The owner can update the maximum transaction amount to any value greater than 1 million tokens.*
- *The owner can exclude/include accounts from the maximum wallet amount and maximum transaction amount restrictions at any time.*
- *The owner can enable/disable the automatic swapping functionality at any time.*
- *The owner can update the threshold number of tokens needed to trigger the automatic swapping functionality to any value between 10,000 tokens and 5 million tokens.*

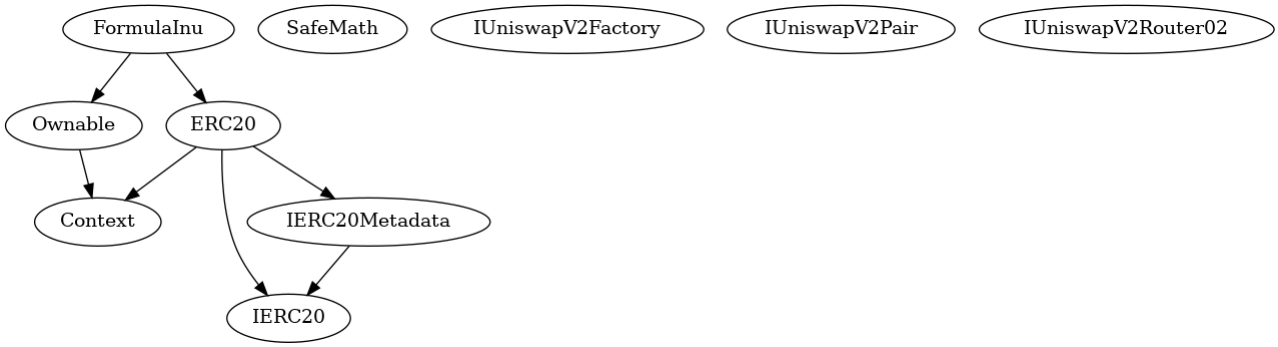
> - *The owner can update the percentage of tokens that are automatically sent to the 0x..dead address from the Uniswap Pair address on sells via Uniswap to any percentage between 0% and 10%.*
> - *The owner can update the amount of time that must pass before the above functionality can reoccur to any time greater than 10 minutes (in seconds).*
> - *The owner can also manually trigger the sending of tokens to the 0x..dead address from the Uniswap Pair once per 30 minutes.*
> - *The owner can disable the transfer delay restriction at any time.*
> - *The owner can withdraw any tokens or ETH from the contract at any time.*
> - *The owner can update the team's Marketing wallet and Dev wallet to any addresses at any time.*
> - *The owner can update the Automated Market Maker Pair address at any time.*

AUDIT RESULTS

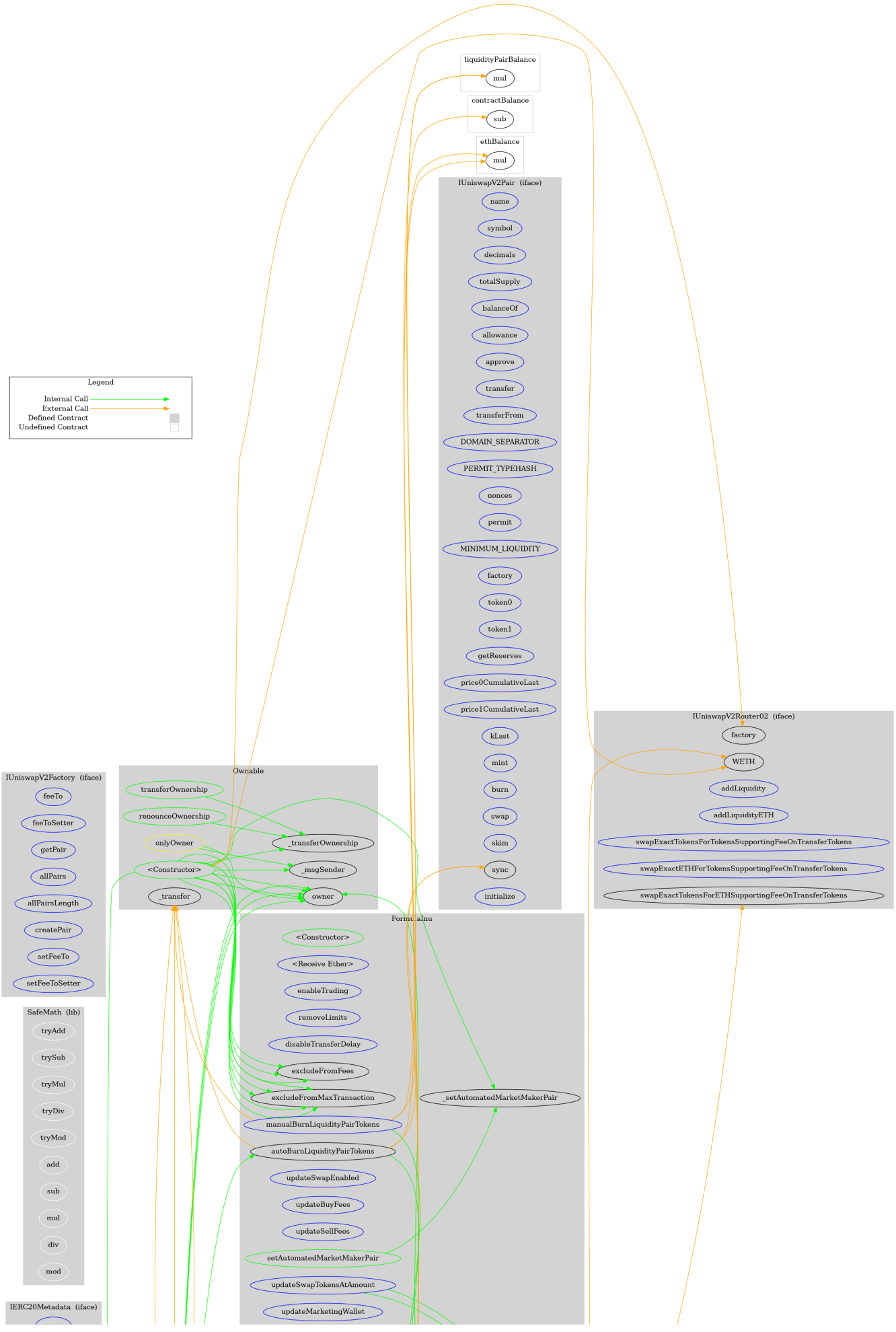
Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	◦ The LP tokens generated through automatic liquidity adds are sent to the owner. ◦ The owner can add accounts to the transfer blacklist at any time. ◦ The owner can withdraw $FINU tokens from the contract at any time. ◦ The owner can manually send up to 10% of the Uniswap Pair token balance to the 0x..dead address every 30 minutes.	WARNING
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	The automatic token burning functionality from the Pair address is susceptible to front running. The team must monitor and if suspicious activity is detected, should either lower the `percentForLPBurn` value or disable this system altogether.	WARNING
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS

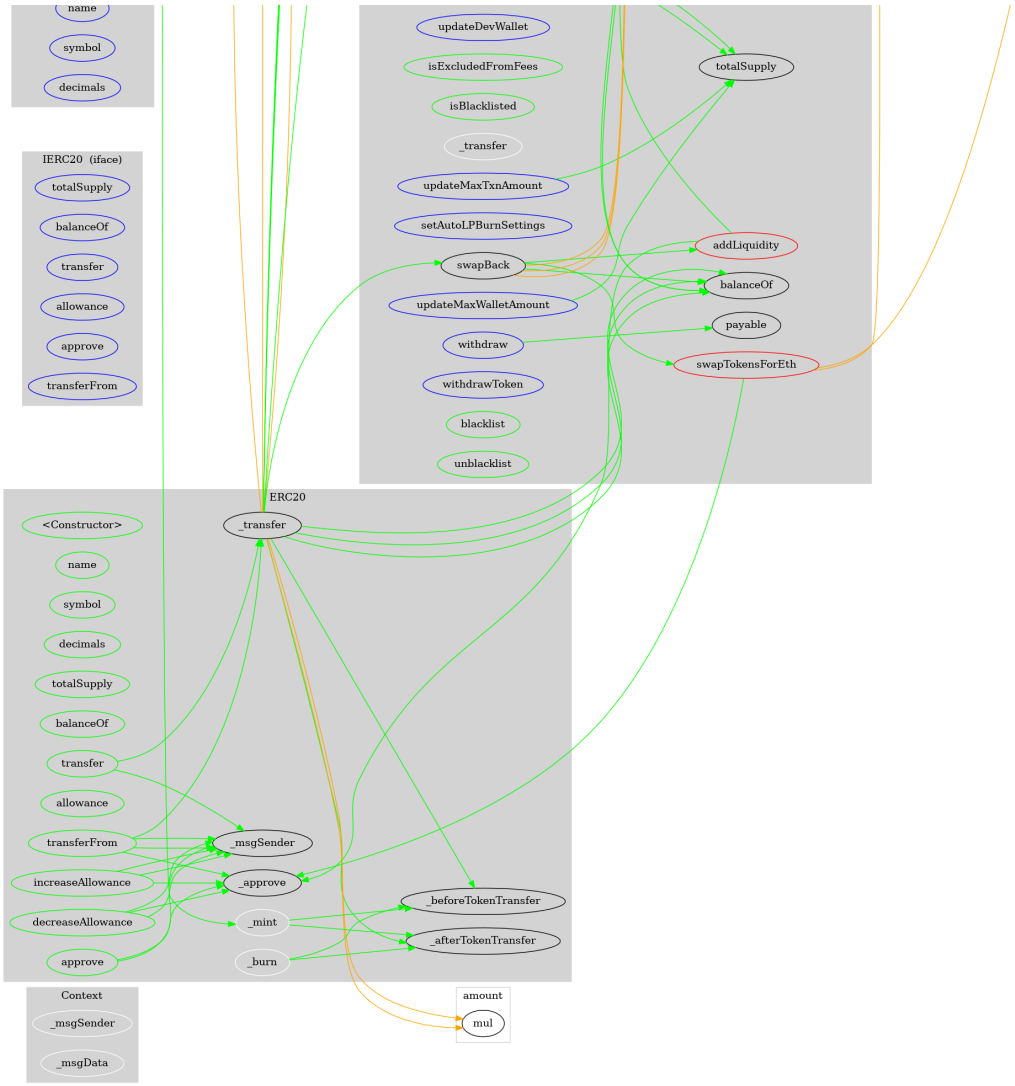
Vulnerability Category	Notes	Result
Signature Issues	N/A	PASS
Sybil Attack	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

INHERITANCE CHART



FUNCTION GRAPH


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
liquidityPairBalance
mul
contractBalance
sub
ethBalance
mul
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token0
token1
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize
IUniswapV2Router02 (iface)
factory
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
Ownable
transferOwnership
renounceOwnership
onlyOwner
<Constructor>
_transfer
_transferOwnership
_msgSender
owner
Formu_aInu
<Constructor>
<Receive Ether>
enableTrading
removeLimits
disableTransferDelay
excludeFromFees
excludeFromMaxTransaction
manualBurnLiquidityPairTokens
autoBurnLiquidityPairTokens
updateSwapEnabled
updateBuyFees
updateSellFees
setAutomatedMarketMakerPair
updateSwapTokensAtAmount
updateMarketingWallet
_setAutomatedMarketMakerPair
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
IERC20Metadata (iface)



FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  Context
   - [Int] _msgSender
   - [Int] _msgData

+  Ownable (Context)
   - [Pub]  #
   - [Pub] owner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner
   - [Int] _transferOwnership #

+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
```

```
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ [Int] IERC20Metadata (IERC20)
   - [Ext] name
   - [Ext] symbol
   - [Ext] decimals

+  ERC20 (Context, IERC20, IERC20Metadata)
   - [Pub]  #
   - [Pub] name
   - [Pub] symbol
   - [Pub] decimals
   - [Pub] totalSupply
   - [Pub] balanceOf
   - [Pub] transfer #
   - [Pub] allowance
   - [Pub] approve #
   - [Pub] transferFrom #
   - [Pub] increaseAllowance #
   - [Pub] decreaseAllowance #
   - [Int] _transfer #
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _approve #
   - [Int] _beforeTokenTransfer #
   - [Int] _afterTokenTransfer #

+ [Lib] SafeMath
   - [Int] tryAdd
   - [Int] trySub
   - [Int] tryMul
   - [Int] tryDiv
   - [Int] tryMod
   - [Int] add
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] mod
   - [Int] sub
   - [Int] div
   - [Int] mod

+ [Int] IUniswapV2Factory
   - [Ext] feeTo
   - [Ext] feeToSetter
   - [Ext] getPair
   - [Ext] allPairs
   - [Ext] allPairsLength
   - [Ext] createPair #
   - [Ext] setFeeTo #
   - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
   - [Ext] name
   - [Ext] symbol
   - [Ext] decimals
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transfer #
```

```
    - [Ext] transferFrom #
    - [Ext] DOMAIN_SEPARATOR
    - [Ext] PERMIT_TYPEHASH
    - [Ext] nonces
    - [Ext] permit #
    - [Ext] MINIMUM_LIQUIDITY
    - [Ext] factory
    - [Ext] token0
    - [Ext] token1
    - [Ext] getReserves
    - [Ext] price0CumulativeLast
    - [Ext] price1CumulativeLast
    - [Ext] kLast
    - [Ext] mint #
    - [Ext] burn #
    - [Ext] swap #
    - [Ext] skim #
    - [Ext] sync #
    - [Ext] initialize #

+ [Int] IUniswapV2Router02
    - [Ext] factory
    - [Ext] WETH
    - [Ext] addLiquidity #
    - [Ext] addLiquidityETH ($)
    - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
    - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
    - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+  FormulaInu (ERC20, Ownable)
    - [Pub]  #
        - modifiers: ERC20
    - [Ext]  ($)
    - [Ext] enableTrading #
        - modifiers: onlyOwner
    - [Ext] removeLimits #
        - modifiers: onlyOwner
    - [Ext] disableTransferDelay #
        - modifiers: onlyOwner
    - [Ext] updateSwapTokensAtAmount #
        - modifiers: onlyOwner
    - [Ext] updateMaxTxnAmount #
        - modifiers: onlyOwner
    - [Ext] updateMaxWalletAmount #
        - modifiers: onlyOwner
    - [Pub] excludeFromMaxTransaction #
        - modifiers: onlyOwner
    - [Ext] updateSwapEnabled #
        - modifiers: onlyOwner
    - [Ext] updateBuyFees #
        - modifiers: onlyOwner
    - [Ext] updateSellFees #
        - modifiers: onlyOwner
    - [Pub] excludeFromFees #
        - modifiers: onlyOwner
    - [Pub] setAutomatedMarketMakerPair #
        - modifiers: onlyOwner
    - [Prv] _setAutomatedMarketMakerPair #
    - [Ext] updateMarketingWallet #
        - modifiers: onlyOwner
    - [Ext] updateDevWallet #
        - modifiers: onlyOwner
    - [Pub] isExcludedFromFees
    - [Pub] isBlacklisted
```

```
    - [Int] _transfer #
    - [Prv] swapTokensForEth #
    - [Prv] addLiquidity #
    - [Prv] swapBack #
    - [Ext] setAutoLPBurnSettings #
       - modifiers: onlyOwner
    - [Int] autoBurnLiquidityPairTokens #
    - [Ext] manualBurnLiquidityPairTokens #
       - modifiers: onlyOwner
    - [Ext] withdraw #
       - modifiers: onlyOwner
    - [Ext] withdrawToken #
       - modifiers: onlyOwner
    - [Pub] blacklist #
       - modifiers: onlyOwner
    - [Pub] unblacklist #
       - modifiers: onlyOwner
```

ABOUT SOLIDITY FINANCE

Solidity Finance was founded in 2020 and quickly grew to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1300+ solidity smart contract audits covering all major project types and protocols, securing a total of over $50 billion U.S. dollars in on-chain value across 1500 projects!.
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.