BlockchainMultiSniper Installation and Setup Guide

About the bot

BlockchainMultiSniper is a bot which automatically buys and sells tokens from Telegram channels capable of operating on over 220 EVM-compatible blockchains and any Uniswap-forked DEX.

This bot is currently in Public Beta at time of writing, which means that there may still be bugs with the bot.

The bot allows you to buy tokens paired with any token (eg. ETH, BNB, USDT). It automatically buys tokens by monitoring messages from an unlimited amount of Telegram channels and getting the contract address from the message. It constantly updates the price and shows you the profit you've made, has the ability to autosell at a specified profit margin (eg, 2x, 10x etc), sell using trailing stop loss and selling after a specified time (eg. 30 seconds).

BlockchainMultiSniper is entirely controlled by the config file, unlike BlockchainLaunchSniper where there is an initial user interface.

BlockchainMultiSniper features:

- Cross-platform (Windows, Mac, Linux etc).
- Supports all EVM-compatible blockchains.
- Scans Telegram messages from any channel and will automatically filter out and buy token from contract address.
- Auto sell with multiple methods (basic autosell, trailing stop loss and selling after time delay).
- Stop loss (sell when initial investment is below certain value eg. 0.5x).
- Honeypot checker.
- Rugpull front-running (available if you have a fast private node).
- Partial sell (eg. sell 10-90% of tokens then sell the rest later when you wish).
- Easy to use config file to change a wide variety of settings eg. wallet address / private key, gas price / limit, node URL, trading strategy etc.
- Price updates multiple times a second showing profit multiplier (eg. 1.5x, 2x), current value of tokens and profit percentage.
- ‘no win no fee’ - only pay 10% fees on profit if you make a profit (less if you have a silver tier or higher - diamond tier has no fees).
- And much more.

By avoiding web interfaces and using nodes directly you can snipe tokens much faster than any of the web-based platforms. This allows tokens to be sniped almost instantly. The bot buys the tokens using the user's wallet address and private key. This information is kept secure, is only stored locally on your computer, and is only ever used to buy / sell tokens.

The bot's source code is heavily compiled, packed and obfuscated to prevent people stealing code or harm the security of this bot. If you have concerns about the security of this bot then you should create a new wallet with a small amount of crypto etc and use that wallet's details in the config file. If you make profit then that can be transferred to your main wallet. You can also use a virtual machine to run the bot.

Another reason the is closed source is that it prevents people from ripping off the code and trying to make / promote their own bot which would cause the developers to lose out on income as funding is important to keep development going and pay expenses etc.

Getting started

To get the bot up and running follow these simple steps.

Prerequisites

- Windows, Mac or Linux computer (windows preferred)
- A reasonably fast internet connection
- Blockchain wallet address and private key (not seed phrase) - recommended to create a fresh wallet on Metamask
- A blockchain node (either use a free node / public RPC URL, or if you want faster results and use rug monitoring then use a private node)
- Enough crypto in your wallet to snipe tokens (and other paired tokens if you choose a different liquidity pair address eg. BUSD)
- Your wallet address must be registered (if not already registered, buy a premium tier from blockchaintokensniper.com/buy)

How to install / run

The BlockchainMultiSniper executable can be supplied with an argument for the config file you wish to use eg. BlockchainMultiSniper ethereum_config.json (all config files must be in .json format).

Windows:

- *Edit the config.json file with your parameters.*
- *(Optional) - if you plan to use the rugpull front-running feature, use the included latency tester tool: run the .py file and the latency tester will measure the average latency of your node (read the README file).*
- *Double-click the .bat file to run (please note that some antivirus systems may detect it as a false positive, this is due to the way it is compiled but the bot is not a threat and is completely safe).*

***Mac OS**:*

- *Edit the config.json file with your parameters.*
- *(Optional) - if you plan to use rugpull front-running feature, use the included latency tester tool: run the .py file and the latency tester will measure the average latency of your node.*
- *Navigate to the executable directory in terminal and run ./BlockchainMultiSniper config.json (replace config.json with the filename of your json config file if needed)*
- *If you have permission issues, go to Security and Privacy in your Mac settings, and allow BlockchainMultiSniper to run.*

Linux:

- *Edit the config.json file with your parameters.*
- *(Optional) - if you plan to use the rugpull front-running feature, use the included latency tester tool: run the .py file and the latency tester will measure the average latency of your node.*
- *Navigate to the executable directory in terminal and run ./BlockchainMultiSniper config.json (replace config.json with the filename of your json config file if needed)*

Config

Note:

Integer: A whole number (eg. 0, 10)
Float: Any number that can include a decimal (eg. 1, 0.1, 105.5)
Boolean: True or False (make sure it is spelt 'True' or 'False' - other spellings will NOT work)
String: A piece of text (strings used in config should not have any quotation marks ' or " in them)

* *Don't edit this value unless you are sure you know what you are doing.*

** *Optional.*

Multi sniper config settings:

"walletAddress" - (string): Your wallet address that you registered online (must start with '0x').

"walletPrivateKey" - (string): Your wallet private key (***not*** seed phrase).

_ _ _ _ _

(Optional) Sniping multiple tokens at the same time, or want to use another wallet to snipe?

The bot has a feature where you can deploy multiple instances of the bot with different wallets at the same time, so you can snipe multiple tokens at the same time. You must still provide your wallet address and private key that you registered with the bot, however if you enable these settings then the bot will use the wallet specified below to snipe. ***If you only snipe with your registered wallet then ignore this section.***

** "activeWalletAddress" - (string): Your wallet address (must start with '0x').

** "activeWalletPrivateKey" - (string): Your wallet private key (***not*** seed phrase).

Example configuration (config.json):

```
{
    "walletAddress": "0x46202b57a5f6a0FF4cE317c976F453185552e2bC",
    "walletPrivateKey": "7495d0cf8087e7571288a104086dda2c2200aa4b14ca38742d07f4905f2d7321",
    "activeWalletAddress": "0x411eC2BcFbec8fb81c9d6aAFE67736832f3d56aD",
    "activeWalletPrivateKey": "2c0b3d2a47ab18a78bd78aa4112a3faaa6878a5e608dde5f1c0e3b71ab8706f1",
    (Other settings).....
}
```

Where *'walletAddress'* would be the wallet you used to register the bot, and *'activeWalletAddress'* would be the wallet that you will actually use for sniping.

_ _ _ _ _

“blockchainNode” - (string): The blockchain node URL you wish to use. Must start with ‘*ws*’, ‘*wss*’, ‘*http*’ or ‘*https*’ prefix, however secure websocket (‘*wss://*’) nodes are recommended.

“exchangeSettings”

This section allows you to change settings for the DEX. You shouldn’t need to edit these settings unless you want to use a new DEX that hasn’t been listed.

* “routerAddress” - (string): the router address for the DEX you wish to use.

* “factoryAddress” - (string): the factory address for the DEX you wish to use.

* “routerABI” - (string): the filename of the ABI for your chosen DEX, eg. ‘PancakeSwap_RouterABI.json’.

* “factoryABI” - (string): the filename of the ABI for your chosen DEX, eg. ‘PancakeSwap_FactoryABI.json’.

* “buyWithCoinFunction” - (string): name of function to buy with blockchain native coin, eg. ‘swapExactETHForTokens’

* “buyWithAltPairFunction” - (string): name of function to buy tokens with alt pair, eg. ‘swapExactTokensForTokens’

* “sellWithCoinFunction” - (string): name of function to sell tokens to blockchain native coin, eg. ‘swapExactTokensForETH’

* “sellWithAltPairFunction” - (string): name of function to sell tokens to alt pair, eg. ‘swapExactTokensForTokens’

“gasSettings”

“buy_gasAmount" - (integer): Gas limit when buying token.

“buy_gasPrice" - (integer): Gas price when buying token.

“approve_gasPrice" - (integer): Gas price for approving - note that there is no standard gas limit for approval TX’s but it is quite low (usually costs about 10-15 cents).

“sell_gasAmount" - (integer): Gas limit when selling token.

“sell_gasPrice" - (integer): Gas price when selling token.

“tradingSettings”

“buyTokens" - (boolean): Buy tokens through the bot? You may wish to disable if you are just selling tokens you’ve bought.

“sellTokens" - (boolean): Sell your tokens through the bot? (recommended).

“tradingMode” - (string): Which trading mode to use to sell tokens. Options:

‘*BASIC_AUTOSELL*’ (sell token at specified price multiplier eg. 2x, 10x etc)
‘*TRAILING_STOP_LOSS*’ (sell token using trailing stop loss: more info below)
‘*SELL_AFTER_TIME_DELAY*’ (sell token after X amount of seconds)

“totalAllowedSnipes” - (integer): Allows you to set a cap on the number of tokens you buy, eg. set to 5 to prevent the bot buying more than 5 tokens. Set to -1 by default to disable (no cap).

“stopLossMultiplier" - (float): Stop-loss multiplier to be used, eg. set to 0.5 to sell when investment goes below 0.5x its initial value (set to 0 to disable).

“autoSellPercentage” - (integer): Percentage of tokens to sell when selling eg. set to 50 to sell 50% of tokens when the bot will attempt to sell.

“allowBuyPreviouslyOwnedTokens” - (boolean): Allow the bot to buy tokens that have been previously bought if the token’s contract address is detected in a TG message? Recommended to disable.

“useSingleLiquidityPair” - (boolean): Using a single liquidity pair? Enable this option to skip checks and potentially buy slightly faster. Please be aware that the bot will not check the token’s liquidity amount if this is disabled.

“transactionRevertTime” - (integer): Time in seconds that a transaction will be processing before it will revert. For big tokens it’s recommended to set quite low eg. 20 seconds so if the transaction doesn’t get processed fast enough it will revert.

“rugMonitoringEnabled” - (boolean): Enable rug monitoring feature? This feature will scan the mempool at all times and frontrun a malicious transaction (eg. remove liquidity, disable trading etc).

This feature will only work if you have a private node.

“basicAutoSell”

“autoSellMultiplier” - (float): Auto sell multiplier for BASIC_AUTOSELL trading mode, eg. set to 2x to make the bot sell when price is or greater than 2x original purchase price.

“trailingStopLoss”

———

What is trailing stop loss?

“A trailing stop is an order type designed to lock in profits / limit losses as a trade moves favourably.”

For example, you could buy a token and sell at 2x the original value.

However, the token may increase in value even more after you sell it, so you could have missed out on alot more profit. If you sold a token at 2x its original value and it reached 10x, you would probably be annoyed you sold too early. So how do we make sure we squeeze the most amount of profit out of tokens?

Trailing stop loss is the solution to this. But first, let’s explain what stop loss is.

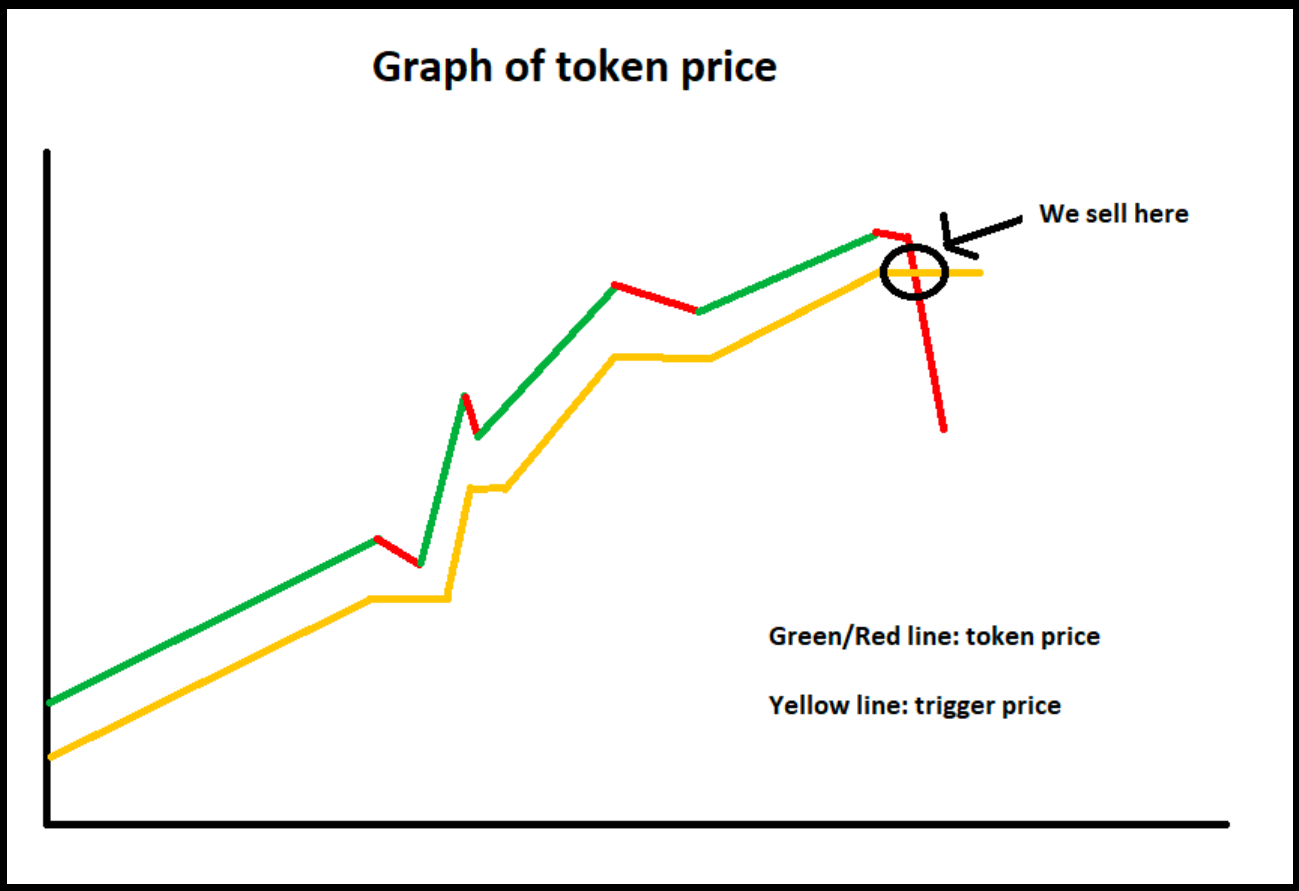
A stop loss is an order to sell your tokens when the value of your tokens goes below a certain value (trigger price) eg. 0.5x.

For example, we buy a token for 1 ETH, but the price keeps going down. Instead of watching the token price tank, we can sell when the price goes to 0.5x (which would be 0.5 ETH).

We will have lost money, but it’s better selling our tokens for 0.5 ETH instead of 0 ETH.

Trailing stop loss is a stop loss, but the trigger price rises as the token price rises. If the token price pumps and then dumps, we can sell at the start of the dump and still make nice profits and sell before the token dumps.

In a trailing stop loss, the trigger price will never decrease. To make it more clear, here's a diagram showing how trailing stop loss works:



As you can see, the bot will sell (stop-out being triggered) as soon as the token price is the same as (or lower than) the trigger price.

You'll notice there's a vertical gap between the token price and the trigger price. This is called the trail amount (trailMultiplierAmount in config).

The trail amount basically means how much leeway we give the token price before we sell.

If we set the trail amount to 0, as soon as the token price dips even slightly, the bot will instantly sell. However, tokens sometimes dip slightly before the price goes up (look at above graph for example), and the bot would sell at the 1st dip.

Increasing the trail amount is often better as the bot will resist the small dips which will make you more profit in the end. Be careful not to set it too big or you may end up losing profit.

For example, imagine we have a token and we set trailMultiplierAmount to 0.5x. The token price rises up to 10x, but then the price tanks. The trigger price will stay at 9.5x (0.5x behind the current price multiplier), so the bot will end up selling at 9.5x.

———

"minTrailingStopLossMultiplier" - (float): Minimum price multiplier before stop-loss will work. Set to 1 by default.

“trailMultiplierAmount” - (float): trail amount measured in price multiplier (eg. 0.5 for 0.5x).

“trailingStopLossHardcap” - (float): Price multiplier hardcap to instantly sell at without further monitoring price while trailing stop loss is enabled. Set to ‘-1’ to disable.

“sellAfterTimeDelay”

“sellAfterSeconds” - (integer): amount of seconds to sell after token is purchased. Eg. Set to 60 to sell exactly one minute after purchase.

“honeypotCheckerSettings”

“allowCheck” - (boolean): enable honeypot check for tokens? (Recommended)

“maxBuyFee" - (integer): Max acceptable buy fee before bot will buy.

“maxSellFee” - (integer): Max acceptable sell fee before bot will buy.

“honeypotCheckerAddress” - (string): Contract address of honeypot checker.

To get honeypot checker addresses, type /dexinfo into either the main group or testing group on Telegram. This will list all DEXs that currently have a honeypot checker address.

“liquidityPairs”

In this section, you can add an unlimited amount of liquidity pairs eg. BNB, BUSD, USDT etc.

Unlike other sections in the config file, this config entry can be repeated an unlimited amount of times for each liquidity pair. The config file has an illustration of how this section could be filled in.

“symbol” - (string): symbol of liquidity paired token eg. BNB, BUSD

“snipeAmount” - (float): amount to buy in liquidity paired token for each liquidity pair eg. if in BUSD section and set to 10, the bot will buy 10 BUSD worth of a token if it is paired with BUSD.

“minLiquidityAmount” - (float): minimum liquidity amount allowed eg. if in BUSD section and set to 1000, the bot will ignore tokens with less than 1000 BUSD liquidity.

“maxLiquidityAmount” - (float): maximum liquidity amount allowed eg. If in BUSD section and set to 1000000, the bot will ignore tokens with more than 1M BUSD liquidity.

NOTE: to disable liquidity amount checks, set both minLiquidityAmount AND maxLiquidityAmount to ‘-1’.

“telegramSettings”

This section relates to your Telegram account.

Before using the bot, you will have to create a Telegram application in my.telegram.org - once completed you will be given an API ID and hash, which is needed for the bot to connect to your Telegram account and read the messages from the Telegram channels you have configured.

The bot only reads messages from the Telegram channels you have configured and does not access or change your account in any other way.

“api_id” - (string): the API ID from your Telegram application.

“api_hash” - (string): the API hash from your Telegram application.

“proxySettings”

A proxy is not needed for the majority of users. However you can use this if you wish / need to.

“enableProxy” - (boolean): Enable Telegram proxy? (Disabled by default)

“proxyType” - (string): Proxy type to use. Options: ‘http’, ‘socks4’, ‘socks5’.

“proxyAddress” - (string): Proxy address to use.

“proxyPort” - (integer): Proxy port to use.

“useProxyCredentials” - (boolean): Use proxy credentials? (Disabled by default)

“proxyUsername” - (string): Username to use for proxy.

“proxyPassword” - (string): Password to use for proxy.

“useProxyRDNS” - (boolean): Use Proxy RDNS? (Disabled by default)

“telegramChannels”

Here you can specify which Telegram channels you would like the bot to monitor and snipe.

Please note that only public Telegram channels are supported at the moment, and you must join the Telegram channel with the Telegram account you use with this bot.

Like the liquidityPairs section above, you can provide an unlimited amount of Telegram channels, as demonstrated in the config file.

“channelName” - (string): The username of the Telegram channel you want to snipe.

The whitelistedText and blacklistedText feature allows you to require and / or prevent the bot from sniping if text is not / is present.

For example, if the bot is configured with:

Whitelisted text: “apple”, “orange”, “banana”

Blacklisted text: “grape”, “mango”, “pear”

And we have the following message:

“apple orange banana 0x0000000000000000000000000000000000000001” - the bot would snipe this as all whitelisted text is present and the contract address is present.

“apple melon banana 0x0000000000000000000000000000000000000001” - the bot would not snipe this as a whitelisted phrase is not provided.

“pineapple orange banana 0x0000000000000000000000000000000000000001” - the bot would snipe this as all whitelisted text is present and the contract address is present.

“apple orange banana pear 0x0000000000000000000000000000000000000001” - the bot would not snipe this as although all whitelisted text is present and the contract address is present, there is blacklisted text.

“apple orange banana” - the bot would not snipe this as there is no contract address to snipe.

“whitelistedText” - (strings): Array of whitelisted texts. Must be written as eg. [‘whitelistedText1’, ‘whitelistedText2’, ‘whitelistedText3’]

“blacklistedText” - (strings): Array of blacklisted texts. Must be written as eg. [‘blacklistedText1’, ‘blacklistedText2’, ‘blacklistedText3’]

“miscellaneous”

“enableConsoleColours” - (boolean): Display colours in your console? Some older terminals may not support console colours so you can disable if needed.

“enableConsolePriceUpdates" - (boolean): Enable cyan console price updates?

“priceUpdateDelayTime” - (integer): Number of seconds between each price update, can be useful if bot is too resource-intensive. Set to 0 to disable (default).

Recommended TG channels

Remember to DYOR (do your own research) before using these Telegram channels. We are not affiliated with them and are not responsible if you lose money.

@CMC_fastest_alerts - posts token before they are published on CoinMarketCap. Recommended to use ‘first pump’ as whitelisted text.

@CG_fastest_alerts - same as above but for CoinGecko.

@bscsafelockcalls - Posts new BSC tokens when liquidity has been added.

Troubleshooting

“Failed to connect to node” - could be a variety of reasons, such as the node not running / broken, your internet connection, firewall etc. Try a different computer / node.

Transaction fails:
- Use dashboard.tenderly.co and paste in the TX hash of the failed transaction.
- You should get an error code in the top left of the screen (red box).
- “BP: address is not whitelisted”: the token you tried to snipe was a whitelist token, you didn’t whitelist your wallet.
- “BEP20: transfer amount exceeds allowance”: check you approved the liquidity paired token before using the bot.
- Other errors could be due to a badly configured token, anti-bot system, broken contract etc.

Incorrectly configured config.json file:

- Use a json formatter / validator (https://jsonformatter.curiousconcept.com is recommended).
- Paste the contents of your config.json file into the validator and click on Process.
- The validator should show you where there is an issue. Make sure that you have correctly used the following symbols: “ ‘ } { [ ]
- Make sure when you save the config.json file that you use the correct encoding (ANSI).

If you have any other issues feel free to ask in the Telegram group.

Strategies

The multi sniper does not guarantee that you will make a profit. However, if you use the right strategy you are very likely to make good profits, and you have a huge advantage over manual buyers.

In general, to make the most profit and snipe the fastest you will need:

- A fast computer / VPS.
- A fast internet connection.
- A private node - recommended to build your own node such as a geth node, or use a Hetzner node for example (AX61-Nvme will work fine although some cheaper ones may work just as well) - see https://docs.binance.org/smart-chain/developer/fullnode.html for more details.

  Public RPC URLs will work fine however if you don't want to use a private node.

- A fairly large amount of crypto to invest in a snipe. You can invest any amount you like, but in general, investing eg. 0.1 BNB or more will yield the most profits.
- High gas fees (specifically a high gas price for buying).

Blockchain validators usually prioritise TX's that pay higher gas, so you will effectively 'jump the queue' when sniping (buying) and get in first so you can purchase for the lowest price, which will give you the best advantage and make you the most profit. Gas fees will be higher but if investing a larger amount of money it it significantly worth it.

In general, a gas price of at least 10 gwei is recommended for buying. However you can go even higher to get in faster, and some bots pay 100 or even 1000+ gwei gas when sniping.
The minimum gas price that will be accepted is 5 gwei. However, it is not recommended to use this low a gas price as you will be at the 'bottom of the pile' and your transaction may end up being processed later than desired, which could cause you to miss out on alot of profit or lose money. Gas fees for selling aren't so important but more than 5 gwei gas is recommended.

Trailing stop loss is recommended when selling to make the most profit.

How to spot / avoid scams

Unfortunately, crypto is full of scams and there are scammers everywhere trying to steal your money.

Scams happen all the time in crypto and are constantly evolving but with some basic knowledge you should be able to avoid the majority of them.

The most common form of scam is a 'rugpull'. This happens when people buy a token but the liquidity is unlocked. This allows the scammer to easily withdraw the liquidity which is where all your funds go. Some scammers make as much as $100k+ in a few minutes with this type of scam so unfortunately it's not going to stop anytime soon.

Another type of scam, often integrated with the rug pull, is a honeypot. This happens when the token developers alter the code of the smart contract so tokens can be bought, but not sold, so effectively your money is stolen as you cannot withdraw it. Most honeypots have trading disabled at the start so as soon as they launch they can be detected, but some honeypots are 'dynamic'

which means that the token developers can disable trading at any point in time. The only way to check this is by analysing the smart contract.

When you buy tokens and have made a profit it is generally advised to sell quickly. These new tokens often end up to be ‘pump and dumps’ which means that the price will rapidly rise at launch and then dump right back to where it started.

Also be aware that new tokens that lock liquidity sometimes lock them for short periods of time (eg. A few days or hours), after which time they could easily rugpull.

The following tools are recommended for detecting and avoiding scams:

- rugpulldetector.com - paste token contract code, will check for any bad functions / scams
- honeypot.is - checks if token is a honeypot *at that moment in time*
- staysafu.org/scanner - a good scanner that checks for many issues
- bscheck.eu - another good scanner that checks for many issues

I’m still stuck with something.

Please look through the BlockchainTokenSniper Telegram group as your question is likely to have been answered already. However if you still can’t find the answer then feel free to ask in the TG group and someone will likely answer your question quickly.

Got any other issues / suggestions / improvements?

Please feel free to mention issues in the group or contact the main dev @BytePhoenix on Telegram.

Telegram group: t.me/blockchaintokensniper

Happy sniping!

BlockchainTokenSniper team.