# Trade Trap

### 1.Introduction

The previous two type of vulnerabilities are always come out from developers’ negligence so that attackers can seek such code and benefit from it. Whatever, to prevent such mistake developers can only pay more attention to code. But unavoidably, some people with bad will always try to treat others. In smart contract, there are some case that code as trap.

As research from ERC20, publicly tradable ERC-20 tokens have considerable high market value. Various exchanges, either centralized (e.g., Binance, Huobi.pro, and OKex) or decentralized (e.g., IDEX, EtherDelta, ForkDelta), provide the marketplace by listing them, especially with high-liquidity ones, for public trading. Evidently, the transparency and security of their corresponding smart contracts is paramount. In practice, there is a de-facto requirement for these contract to be publicly verifiable on etherscan.io. Moreover, reflecting the fundamental ‘code-is-law’ spirit and trust of blockchain technology, these contracts once deployed should not be further subject to centralized control or manipulation.” Once smart contracts of publicly tradable ERC-20 tokens are deployed, they should not be further subject to centralized control or manipulation. Unfortunately, tradeTrap plagues 700+ ERC20 tokens and we have so far confirmed at least dozens of them are publicly tradable on current exchanges

The existence of highly manipulatable interfaces (or knobs), however, could be exploited to either make inappropriate arbitrage or even directly control buy / sell prices of affected tokens. All these will eventually result in financial loss for trading customers and essentially reflect lack of enough security of affected exchanges when listing thesae tokens for trading.

In the following, we would like to disclose two types of manipulatable interfaces which could be exploited to achieve unfair arbitrage.

### 2. Arbitrary Increase in Token Balance

### 2.1Real world case

Firstly, we would like to expose an interface which can be used to arbitrarily increase the token balance of arbitrary address, completely at the control of contract owner! As shown in Figure 1, a function named mintToken() is implemented in these ERC20 smart contracts.

In general, this function is accessible only by the contract owner, which is used to increase the token supply. However, in spite of a reasonable functionality during the presale period, the invocation to this function should be constrained in some way to avoid any abuse. Otherwise, a malicious owner is capable of sending tokens to a specified addresses arbitrarily. These tokens issued with zero-cost could completely destroy the market at the cryptocurrency exchanges.

### 3. Manipulatable Prices and Unfair Arbitrage

Secondly, we would like to expose a series of functions that can be directly used to achieve arbitrage. In essence, there exists three manipulatable interfaces, defined in corresponding functions i.e., setPrices(), buy() and sell() (Figure 2). The setPrices() function, protected by the onlyOwner modifier, merely allows the owner to set the buy / sell prices of the tokens. The buy() and sell() functions are public accessible and can be used by any trading customers to buy some tokens or sell the tokens they owned.

A keen reader may notice that the aforementioned buy / sell prices seem a little bit weird, as the prices of a tradable token should be determined by the market (i.e., exchanges). Here comes the tricky but interesting part: anyone can be an arbitrager of these tokens if she wants. Here are simple steps she may want to take:

1. One can make a profit by buying tokens with *buyPrice* and then sell them with the market selling price if the latter is *greater* than the former;
2. One can make a profit by buying tokens with the market buying price and then sell them with the *sellPrice* if the latter is *greater* than the former;

As it is impossible to guarantee that the buy / sell prices are identical to their counterparts (i.e., the market prices) all the time, one can always find an opportunity to make (inappropriate) arbitrage. As a result, the circulation of the tokens would not be maintained anymore. Moreover, as both buyPrice and sellPrice are modifiable by the owner, the owner becomes the one to directly control the market price!