SECURITY AUDIT OF

DEXYNTH SMART CONTRACTS

Public Report

Apr 20, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.
BSC	Binance Smart Chain or BSC is an innovative solution for introducing interoperability and programmability on Binance Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Apr 20, 2023. We would like to thank the Dexynth for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Dexynth Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team found some vulnerabilities in the application. Dexynth team has resolved and updated the issue following our recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Dexynth Smart Contracts

Dexynth is a DEFI synthetics trading platform which provides stock trading, forex, commodities, crypto and more.

It aims to deliver a superior, reliable, and equitable leveraged trading experience on a fully decentralized protocol:

- Synthetic Trading
- Wide range of synthetic trading pairs to choose from
- High leverage available
- Custom Chainlink Decentralized Oracle Network
- Guaranteed Stop-Loss. High liquidation range

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Dexynth Smart Contracts.

It was conducted on commit `f9780ec12e907d1d8036c21020fc964ca109b4ac` from git repository link: *https://github.com/Global-Repo/Synths*.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
ce4fd69a77b7d442db5d0719b482226de3ae7df24e374b2086f5285ad2f9183a	V6.3/Tokens/GToken.sol
b86f0860a7042a490730942cf842c7aa3a4ca01d09d5be41a47509e79a019e46	V6.3/GNSTradingCallbacksV6_3.sol
3d3e63e1a72adcf2b7af3abd201a7d86eb16f8b4f477ca85d9a485a207ce1dcc	V6.3/GNSPriceAggregatorV6_3.sol
e17ffc2db9278e5e5a6f916f42250b9992f2846ec8d6dfb9db2e9ed9f296afdc	FuckiesTrade.sol
6ec0bf5662c61f315220063c114909c42cc66a8d0191af486c06d198edabad88	GNSPriceAggregatorV6_2.sol
5102bd4ea86c243de750e97726313113ae7dfb8bc9fe77485ab786f626fc77fe	GNSTradingV6_2.sol

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.

- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract.

However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The Dexynth Smart Contracts was written in `Solidity` language. The source code was written based on OpenZeppelin's library.

2.1.1. GToken contract

GToken is an `ERC20` token contract which acts as a mechanism of liquidity efficiency. It supports the liquidity efficiency of the USDT vault by minting rewards for NFT bots and affiliates.

The `USDT vault` acts as the counterpart in all trades conducted on the platform. Winning trades result in payouts from the vault to the user, while losing trades result in the user making a payment to the vault in `USDT`. Therefor, incentives for liquidity providers are generated by the trading platform and the APR is directly proportional to the trading volume.

2.1.2. GNSTradingV6_2 contract

This contract is the place where users make decentralized trading on the platform. Trades are opened with `USDT` collateral, regardless of the trading pair. The leverage is synthetic and backed by the `USDT vault` and the `GToken`. Whenever a trader earns a positive PNL, `USDT` is taken from the `USDT vault` to pay them, whereas negative PNL trades result in the trader pays `USDT` to the `USDT vault`. The trading contract uses a custom real-time `Chainlink` decentralized oracle network to get the median price for each trading order.

Governance has access to a function that enables them to pause the opening of new trades, which is often utilized when a contract is upgraded. However, this function does not close any traders' open positions, and all traders retain control of closing their trades. Also, there is another one controlled by governance to prevent any interaction with the contract.

2.1.3. FuckiesTrade contract

This is the contract where users use their `Fuckies` tokens to make free trade (open a long/short position with predefined TP/SL). If the trade results in a win, users receive a reward.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of Dexynth Smart Contracts.

Dexynth team fixed the code, according to Verichains's draft report in commit `c320ffb1dcc6af1a2a9e7cb1c8d7339651eb5b87`.

#	Issue	Severity	Status
1	Trade info is overwritten `finalizeTrade` function	CRITICAL	Fixed
2	TP and SL should not be controlled by the user's input price	HIGH	Fixed
3	Users will lose free trade ticket when order is cancelled	MEDIUM	Fixed
4	Order will not be fulfilled if `minAnswers` or `pairNoMedian` is updated while waiting for answers	MEDIUM	Fixed
5	Owner could be Admin while initializing	LOW	Fixed

2.2.1. Trade info is overwritten CRITICAL

Affected files:

- FuckiesTrade.sol
- GNSTradingCallbacksV6_3.sol
- GFarmTradingStorageV5.sol

In `startFreeTradeCallback` function, `TradeInfoHashToTrader` stores mapping from hash of `(pairIndex,tradeIndex)` to `trader`.

```
function startFreeTradeCallback(uint _orderId, uint _pairIndex, uint _tradeIndex) external 
onlyCallbacks {
    // Recovering trader address
    address trader = orderIdToTrader[_orderId];
    // Storing trade
    trades[trader].push(TradeInfo(_pairIndex, _tradeIndex));
    // Update wallet daily trades
    tradesToday[trader] += 1;
    // Update openedTrades
    openedTrades[trader] +=1;
    // Update total daily trades
    totalTradesToday++;
    // Calculate TradeInfoHash
    TradeInfoHashToTrader[keccak256(abi.encode(TradeInfo(_pairIndex, _tradeIndex)))] = 
trader;
    // Store OrderId & Trade
    TradeInfoHashToOrderId[keccak256(abi.encode(TradeInfo(_pairIndex, _tradeIndex)))] = 
_orderId;
    // Emit the event
    emit FuckiesFreeTradeOpened(trader, _orderId, _pairIndex, _tradeIndex);
}

function openTradeMarketCallback(
    AggregatorAnswer memory a
) external onlyPriceAggregator notDone{
    ...
    (StorageInterfaceV5.Trade memory finalTrade, uint tokenPriceDai) = registerTrade(
        t, 1500, 0
    );
    ...
    if (t.trader == address(fuckiesTrade)) fuckiesTrade.startFreeTradeCallback(a.orderId,
finalTrade.pairIndex, finalTrade.index);
    ...
}

function registerTrade(
    StorageInterfaceV5.Trade memory trade,
    uint nftId,
    uint limitIndex
) private returns(StorageInterfaceV5.Trade memory, uint){
    ...
    trade.index = storageT.firstEmptyTradeIndex(trade.trader, trade.pairIndex);
    ...
}
```

The problem is `tradeIndex` is calculated from `firstEmptyTradeIndex` and it returns the first empty trade slot from 0 to `maxTradesPerPair` in `openTrades`. However, it only functions properly if the number of trades of a user in a pair is lower than `maxTradesPerPair`. Otherwise, `firstEmptyTradeIndex` will return default value (0) for `tradeIndex`. Unfortunately, this condition has been disabled for `startFreeTrade` to allow `FuckiesTrade` to perform free trades on behalf of users. As a result, `TradeInfoHashToTrader` and `openTrades` of the first user will be overwritten by the last one whenever there are more open trades than `maxTradesPerPair` (default set to 3) at the same time.

```
// This function will return default value (0) when there is no room for new trade
function firstEmptyTradeIndex(address trader, uint pairIndex) public view returns(uint
index){
    for(uint i = 0; i < maxTradesPerPair; i++){
        if(openTrades[trader][pairIndex][i].leverage == 0){ index = i; break; }
    }
}

function startFreeTrade(StorageInterfaceV5.Trade memory t) external onlyFuckiesTrade
returns(uint) {

    require(!isPaused, "PAUSED");

    AggregatorInterfaceV6_2 aggregator = storageT.priceAggregator();
    PairsStorageInterfaceV6 pairsStored = aggregator.pairsStorage();

    address sender = _msgSender();
    // require(storageT.openTradesCount(sender, t.pairIndex)
    // + storageT.pendingMarketOpenCount(sender, t.pairIndex)
    // + storageT.openLimitOrdersCount(sender, t.pairIndex)
    //     < storageT.maxTradesPerPair(),
    //     "MAX_TRADES_PER_PAIR");

    // require(storageT.pendingOrderIdsCount(sender)
    //     < storageT.maxPendingMarketOrders(),
    //     "MAX_PENDING_ORDERS");
    ...
}
```

RECOMMENDATION

You can either restrict the number of trades that `FuckiesTrade` is permitted to execute or allow for an unrestricted number of trades.

```
// Store fuckiesTradeContract address in GFarmTradingStorageV5.sol
function firstEmptyTradeIndex(address trader, uint pairIndex) public view returns(uint index){
    // Allow for an unrestricted number of trades but it would require a significant amount of gas, particularly when dealing with a huge volume of trades.
    // Alternatively, to mitigate this risk, a safer approach would be to limit the number of trades for the FuckiesTrade contract in both the startFreeTrade and this function.
    uint maxTrade = trader == fuckiesTradeContract ? type(uint256).max : maxTradesPerPair;
    for(uint i = 0; i < maxTrade; i++){
        if(openTrades[trader][pairIndex][i].leverage == 0){ index = i; break; }
    }
}
```

UPDATES

- *Apr 13, 2023*: This issue has been acknowledged and fixed by the Dexynth team.

2.2.2. TP and SL should not be controlled by the user's input price HIGH

Affected files:

- FuckiesTrade.sol
- GNSTradingCallbacksV6_3.sol

The `FuckiesTrade` contract allows users to open free trades with a position size of $50 and `x100` leverage for each trade. The TP and SL are calculated based on their input price, which are 2% and 0.25% respectively. The position will be closed when the trade price hits the TP or SL, if the pnl is greater than $50 (profit 1% based on the leveraged position size), the user will receive a `payoutAmount` of $20. Otherwise, the trade will be considered as losing, and the user will get nothing.

```
function closeFreeTradeCallback(uint _toPay, uint _pairIndex, uint _tradeIndex) external
onlyCallbacks {
    // Find trader address
    address trader = TradeInfoHashToTrader[keccak256(abi.encode(TradeInfo(_pairIndex,
_tradeIndex)))];
    // Find orderId
    uint orderId = TradeInfoHashToOrderId[keccak256(abi.encode(TradeInfo(_pairIndex,
_tradeIndex)))];
    // Update openedTrades
    if (openedTrades[trader] > 0) openedTrades[trader] -= 1;
    // Add winnings to trader balance and store trade result
    bool win = (_toPay > tradeAmount) ? true : false;
    if (win) {
        BUSDToClaim[trader] += payoutAmount;
        winningTrades += 1;
        winningAmount += _toPay - tradeAmount;
    }
    else {
        losingTrades += 1;
        losingAmount += tradeAmount - _toPay;
    }
    // ...
}
```

However, the TP and SL can be controlled by the user by inputting a lower price compared to the actual market price in case of long position, and vice versa. In that way, the TP/SL can be adjusted from 2%/0.25% to 1%/1.25% so that the probability of winning the trade will be higher. Additionally, with each losing trade, the contract will lose more than the expected value (from 0.25% to 1.25%).

```
function startFreeTrade(uint _pairIndex, uint _marketPrice, bool _long) public isNotPaused
{
    // ...
    StorageInterfaceV5.Trade memory tradeConditions;
    tradeConditions.trader = address(this);
    tradeConditions.pairIndex = _pairIndex;
    tradeConditions.index = 0;
    tradeConditions.initialPosToken = 0;
    tradeConditions.positionSizeDai = tradeAmount;
    tradeConditions.openPrice = _marketPrice;
    tradeConditions.buy = _long;
    tradeConditions.leverage = 100;
    tradeConditions.tp = _long ? _marketPrice+(((_marketPrice/100)*tpPercent)/10000000000)
: _marketPrice-(((_marketPrice/100)*tpPercent)/10000000000); // +- x%
    tradeConditions.sl = _long ? _marketPrice-(((_marketPrice/100)*slPercent)/10000000000)
: _marketPrice+(((_marketPrice/100)*slPercent)/10000000000); // +- x%
    // Increment trades counter
    totalRequestedTrades += 1;
    // Request Trade
    uint orderId = tradingContract.startFreeTrade(tradeConditions);
    // ...
}
```

There are also some restrictions for the input price in the `openTradeMarketCallback` function of the `GNSTradingCallbacksV6_3` contract. However, it will not affect much as long as the following conditions are met.

```
// o.wantedPrice is the input price
// t.openPrice is the actual market price after impact (returned from the oracle)
// maxSlippage = o.wantedPrice in this case (100% slippage)
t.buy ? t.openPrice > o.wantedPrice + maxSlippage :
    t.openPrice < o.wantedPrice - maxSlippage) == false

// ===> It means the input price much be greater than 0.5 * t.openPrice
```

RECOMMENDATION

The TP and SL should be calculated from the oracle price in the `openTradeMarketCallback` function of the `GNSTradingContractV6_3` contract.

UPDATES

- *Apr 13, 2023*: This issue has been acknowledged and fixed by the Dexynth team.

2.2.3. Users will lose free trade ticket when order is cancelled MEDIUM

Affected files:

- GNSTradingCallbacksV6_3.sol
- FuckiesTrade.sol

In `openTradeMarketCallback` function, if the order is cancelled either because of the contract is paused or the price from oracle is 0 (which is not users fault). The `FuckiesTrade` users will lose their free trade ticket without opening a trade.

```
function openTradeMarketCallback(
    AggregatorAnswer memory a
) external onlyPriceAggregator notDone{
    ...
    if(isPaused || a.price == 0
    || (t.buy ?
    t.openPrice > o.wantedPrice + maxSlippage :
    t.openPrice < o.wantedPrice - maxSlippage)
    || (t.tp > 0 && (t.buy ?
    t.openPrice >= t.tp :
    t.openPrice <= t.tp))
    || (t.sl > 0 && (t.buy ?
    t.openPrice <= t.sl :
    t.openPrice >= t.sl))
```

```
        || !withinExposureLimits(t.pairIndex, t.buy, t.positionSizeDai, t.leverage)
        || priceImpactP * t.leverage > pairInfos.maxNegativePnlOnOpenP()){

        uint devGovFeesDai = storageT.handleDevGovFees(
            t.pairIndex,
            t.positionSizeDai * t.leverage,
            true,
            true
        );

        storageT.transferDai(
            address(storageT),
            t.trader,
            t.positionSizeDai - devGovFeesDai
        );

        emit DevGovFeeCharged(t.trader, devGovFeesDai);

        emit MarketOpenCanceled(
            a.orderId,
            t.trader,
            t.pairIndex
        );

    }else{
        (StorageInterfaceV5.Trade memory finalTrade, uint tokenPriceDai) = registerTrade(
            t, 1500, 0
        );

        emit MarketExecuted(
            a.orderId,
            finalTrade,
            true,
            finalTrade.openPrice,
            priceImpactP,
            finalTrade.initialPosToken * tokenPriceDai / PRECISION,
            0,
            0
        );

        ///////////////////
        // FUCKIES PROMO //
        ///////////////////

        if (t.trader == address(fuckiesTrade))
fuckiesTrade.startFreeTradeCallback(a.orderId, finalTrade.pairIndex, finalTrade.index);
    }
    storageT.unregisterPendingMarketOrder(a.orderId, true);
}
```

RECOMMENDATION

Add a callback to refund free trade ticket to user when the order is cancelled.

UPDATES

- *Apr 20, 2023*: This issue has been acknowledged and fixed by the Dexynth team.

2.2.4. Order will not be fulfilled if `minAnswers` or `pairNoMedian` is updated while waiting for answers MEDIUM

Affected files:

- GNSPriceAggregatorV6_3.sol

While waiting for answers from oracles, if `minAnswersCalculated` is updated to lower than current `answers.length` by updating either `minAnswers` or `pairNoMedian`, the order will then never be fulfilled even if enough answers have been received to satisfy the updated `minAnswersCalculated` condition. For example, when the `minAnswersCalculated` is 4 and there are already 3 answers. If the `minAnswers` is now set to 2, despite meeting the minimum requirement, this order will never be fulfilled when there are more answers arrive because the check `answers.length == minAnswersCalculated` will never be true (unless `minAnswers` is updated again).

```
function fulfill(
        bytes32 requestId,
        uint price
) external recordChainlinkFulfillment(requestId){
    ...
    // Medians
    if (pairNoMedian[r.pairIndex]) minAnswersCalculated = 1;
    else minAnswersCalculated = minAnswers;

    if(price == 0
        || (price >= feedPrice ?
        price - feedPrice :
        feedPrice - price
        ) * PRECISION * 100 / feedPrice <= f.maxDeviationP){

        answers.push(price);

        if(answers.length == minAnswersCalculated){
            CallbacksInterfaceV6_2.AggregatorAnswer memory a;

            a.orderId = orderId;
            a.price = median(answers);
            a.spreadP = pairsStorage.pairSpreadP(r.pairIndex);

            CallbacksInterfaceV6_2 c = CallbacksInterfaceV6_2(storageT.callbacks());

            if(r.orderType == OrderType.MARKET_OPEN){
                c.openTradeMarketCallback(a);

            }else if(r.orderType == OrderType.MARKET_CLOSE){
                c.closeTradeMarketCallback(a);

            }else if(r.orderType == OrderType.LIMIT_OPEN){
                c.executeNftOpenOrderCallback(a);

            }else if(r.orderType == OrderType.LIMIT_CLOSE){
                c.executeNftCloseOrderCallback(a);

            }else{
                c.updateSlCallback(a);
            }

            delete orders[orderId];
            delete ordersAnswers[orderId];
        }

        emit PriceReceived(
            requestId,
            orderId,
            msg.sender,
            r.pairIndex,
            price,
            feedPrice,
            r.linkFeePerNode
        );
    }
}
```

RECOMMENDATION

Change the condition from `answers.length == minAnswersCalculated` to `answers.length >= minAnswersCalculated`.

UPDATES

- *Apr 13, 2023*: This issue has been acknowledged and fixed by the Dexynth team.

2.2.5. Owner could be Admin while initializing LOW

Affected files:

- GToken.sol

While initializing `GToken`, the require condition `_contractAddresses.owner != _contractAddresses.manager && _contractAddresses.manager != _contractAddresses.admin` will let `owner` and `admin` to be the same address which contradicts the behavior of `updateAdmin`, which forbids this scenario from occurring.

```
function initialize(
    string memory _name,
    string memory _symbol,
    ContractAddresses memory _contractAddresses,
    uint _MIN_LOCK_DURATION,
    uint _maxAccOpenPnlDelta,
    uint _maxDailyAccPnlDelta,
    uint[2] memory _withdrawLockThresholdsP,
    uint _maxSupplyIncreaseDailyP,
    uint _lossesBurnP,
    uint _maxGnsSupplyMintDailyP,
    uint _maxDiscountP,
    uint _maxDiscountThresholdP
) external initializer {

    require(_contractAddresses.asset != address(0)
    && _contractAddresses.owner != address(0)
    && _contractAddresses.manager != address(0)
    && _contractAddresses.admin != address(0)
    && _contractAddresses.owner != _contractAddresses.manager
    && _contractAddresses.manager != _contractAddresses.admin
    && _contractAddresses.gnsToken != address(0)
    && _contractAddresses.pnlHandler != address(0)
    && _contractAddresses.openTradesPnlFeed != address(0)
    && _contractAddresses.gnsPriceProvider.addr != address(0)
    && _contractAddresses.gnsPriceProvider.signature.length > 0
    && _maxDailyAccPnlDelta >= MIN_DAILY_ACC_PNL_DELTA
    && _withdrawLockThresholdsP[1] > _withdrawLockThresholdsP[0]
    && _maxSupplyIncreaseDailyP <= MAX_SUPPLY_INCREASE_DAILY_P
    && _lossesBurnP <= MAX_LOSSES_BURN_P
    && _maxGnsSupplyMintDailyP <= MAX_GNS_SUPPLY_MINT_DAILY_P
    && _maxDiscountP <= MAX_DISCOUNT_P
        && _maxDiscountThresholdP >= 100 * PRECISION, "WRONG_PARAMS");
    ...
}

function updateAdmin(address newValue) external onlyManager{
    require(newValue != address(0), "ADDRESS_0");
    require(newValue != owner() && newValue != manager, "WRONG_VALUE");
    admin = newValue;
    emit AddressParamUpdated("admin", newValue);
}
```

RECOMMENDATION

Add `_contractAddresses.owner != _contractAddresses.admin` condition to require statement.

UPDATES

- *Apr 13, 2023*: This issue has been acknowledged and fixed by the Dexynth team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Apr 20, 2023*	Public Report	Verichains Lab

Table 2. Report versions history