AUDITEK SECURITY

COMPLETE AUDIT REPORT

Hero Battle

Security Assessment

AUGUST 21TH, 2021

Prepared by
ELLEN DOWNING

Approved by
LILLIAN PRATT

Disclaimer

AuditeK. reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. AuditeK.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

AuditeK.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. AuditeK Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

AuditeK.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. AuditeK’s position is that each company and individual are responsible for their own due diligence and continuous security. AuditeK in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	18. August 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://herobattle.app/

Telegram
https://t.me/HeroBattleBSC
https://t.me/HeroBattleBSCNews

Twitter
https://twitter.com/HeroBattleBSC

Facebook
https://www.facebook.com/HeroBattle-468380853567164/

Instagram
https://www.instagram.com/herobattle_bsc/

Youtube
https://www.youtube.com/channel/UCnR7_CAGwsvc3Ewx1LvnbEA

Reddit
https://www.reddit.com/user/HeroBattle

Medium
https://medium.com/@herobattle

Description

HEROBATTLE will be the most attractive game project on BSC with professional game developers combined with blockchain game platform development team. Players will experience a PLAY-TO-EARN RPG model. The difference between HEROBATTLE and a traditional game is the Blockchain economic design used to reward our players for their contributions.

Project Engagement

During the 15th of August 2021, **Hero Battle Team** engaged AuditeK.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. **Hero Battle Team** provided AuditeK.io with access to their code repository and whitepaper.

Logo

Contract Link

https://bscscan.com/address/0xD5c213E95249E3D570E2aCA31a72fB4474D4043b#code

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9-10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7–8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4–6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2–3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0–1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to AuditeK to make sure we understand the size, scope, and functionality of the smart contract.

    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to AuditeK describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

- OpenZeppelin
  - Address
  - Ownable
  - SafeMath
  - SafeMathUint
  - SafeMathInt
  - IterableMapping
  - ERC20
  - IERC20
- Uniswap
  - UniswapV2Factory
  - UniswapV2Pair
  - UniswapV2Router01
  - UniswapV2Router02

Tested Contract Files

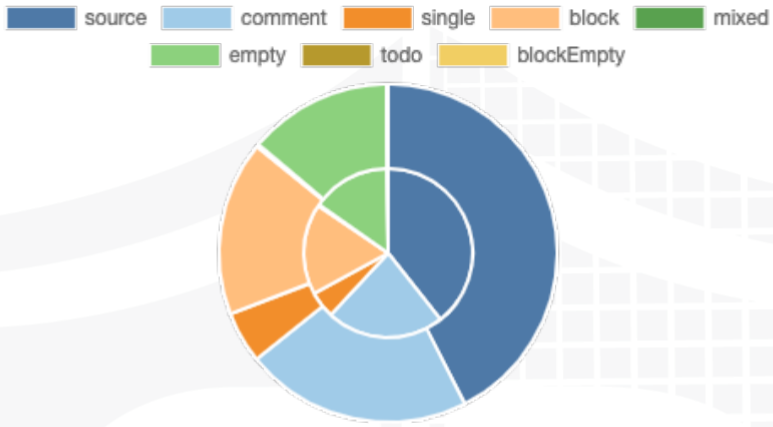
This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

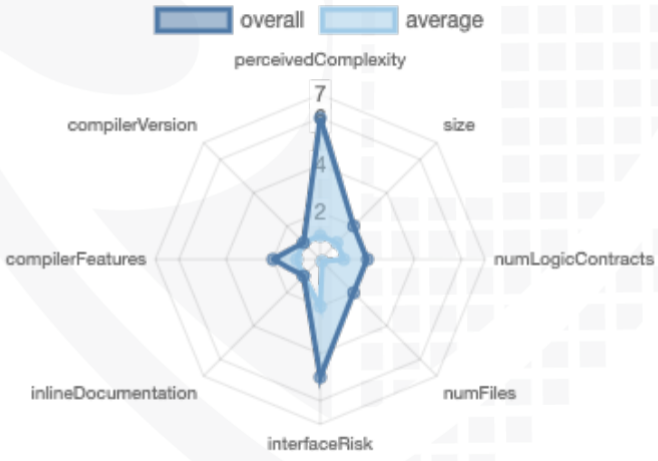
File Name	SHA-1 Hash
contracts/IERC20Metadata.sol	c98275f7bd2c6cb0eaef0dab83be92848ee3c09b
contracts/DividendPayingToken.sol	82c11e2c0a4fc6ef445be6c6e83264f80b41088b
contracts/IUniswapV2Pair.sol	a345c94b2ca004ec5d68ea2fa95dc90d7c0b7e7a
contracts/SafeMathUint.sol	a4719406da9f9075421f53e2b82a916ff9ecaad8
contracts/Context.sol	d5f8c0eeac877f69ba9ada0f9925abecab9acdae
contracts/IUniswapV2Factory.sol	6dfd768eef85f94ca3bf7669a26ff22a8be8ea2e
contracts/DividendPayingTokenInterface.sol	4cd1b102451c554b62939ff3c289d5b7076262ae
contracts/SafeMath.sol	47a55484f7d77a847c0d2638da16315e6404c85c
contracts/IUniswapV2Router.sol	2c8abb8b8215d88897e9011ec4b03e5f4243c71f
contracts/SafeMathInt.sol	a0dc6f88e3a6ae8d51b551ac37b8cd93a53f2545
contracts/Ownable.sol	36cff8d460e596f55fcd07c93f6540e77f94a49f
contracts/IterableMapping.sol	1887d8153ee4f844e42b669ad50069f9310a5fcc
contracts/hero_battle.sol	960e4a20b049cdbb3e0d9fa1f288a341876fe11b
contracts/ERC20.sol	35a78d307b1936c2ddc0cb609ffa1e8d15ae5ccf
contracts/DividendTracker.sol	c7bff1efbd3ccbb0fc4f59f5e2f186b89e0bbbf8
contracts/DividendPayingTokenOptionalInterface.sol	5feafe9dfe56ab29f8cca48beac4f1d75b15565e
contracts/IERC20.sol	1bdf256058bece58d0b44372a3e77d6d5df48cc3

Metrics

Source Lines



Risk Level



Capabilities

Components

Contracts	Libraries	Interfaces	Abstract
5	4	8	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Public	Payable
143	5

External	Internal	Private	Pure	View
92	119	7	25	60

State Variables

Total	Public
36	21

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.6.2`		`yes`	**** (0 asm blocks)	

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/ Create/ Create2

yes					yes → NewContract: DividendTracker

Scope of Work

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Optional implementations

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	✓	✓	✗

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✓	Main
Comment	Line: -			

Max / Total Supply:

```
constructor() public ERC20("HEROBATTLE TOKEN", "HRB") {

    dividendTracker = new DividendTracker();
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
     // Create a uniswap pair for this new token
    address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
         .createPair(address(this), _uniswapV2Router.WETH());

    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair = _uniswapV2Pair;

    _setAutomatedMarketMakerPair(_uniswapV2Pair, true);

    // exclude from receiving dividends
    dividendTracker.excludeFromDividends(address(dividendTracker));
    dividendTracker.excludeFromDividends(address(this));
    dividendTracker.excludeFromDividends(owner());
    dividendTracker.excludeFromDividends(deadWallet);
    dividendTracker.excludeFromDividends(address(_uniswapV2Router));

    // exclude from paying fees or having max transaction amount
    excludeFromFees(owner(), true);
    excludeFromFees(_marketingWalletAddress, true);
    excludeFromFees(address(this), true);

    /*
        _mint is an internal function in ERC20.sol that is only called here,
        and CANNOT be called ever again
    */
    _mint(owner(), 100000000 * (10**18));
}
```

```
function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }
```

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✓

1. approve →
2. blacklistAddress →
3. claim →
4. decreaseAllowance →
5. excludeFromDividends →
6. excludeFromFees →
7. excludeMultipleAccountsFromFees →
8. increaseAllowance →
9. processDividendTracker →
10. renounceOwnership →
11. setAutomatedMarketMakerPair →
12. setEnableFee →
13. setFee →
14. setMarketingWallet →
15. setSwapTokensAtAmount →
16. setTokenAddress →
17. setTokenForMarketing →
18. transfer →
19. transferFrom →
20. transferOwnership →
21. updateClaimWait →
22. updateDividendTracker →
23. updateGasForProcessing →
24. updateUniswapV2Router →

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

1. approve →
2. blacklistAddress →
3. claim →
4. decreaseAllowance →
5. excludeFromDividends →
6. excludeFromFees →
7. excludeMultipleAccountsFromFees →
8. increaseAllowance →
9. processDividendTracker →
10. renounceOwnership →
11. setAutomatedMarketMakerPair →
12. setEnableFee →
13. setFee →
14. setMarketingWallet →
15. setSwapTokensAtAmount →
16. setTokenAddress →
17. setTokenForMarketing →
18. transfer →
19. transferFrom →
20. transferOwnership →
21. updateClaimWait →
22. updateDividendTracker →
23. updateGasForProcessing →
24. updateUniswapV2Router →

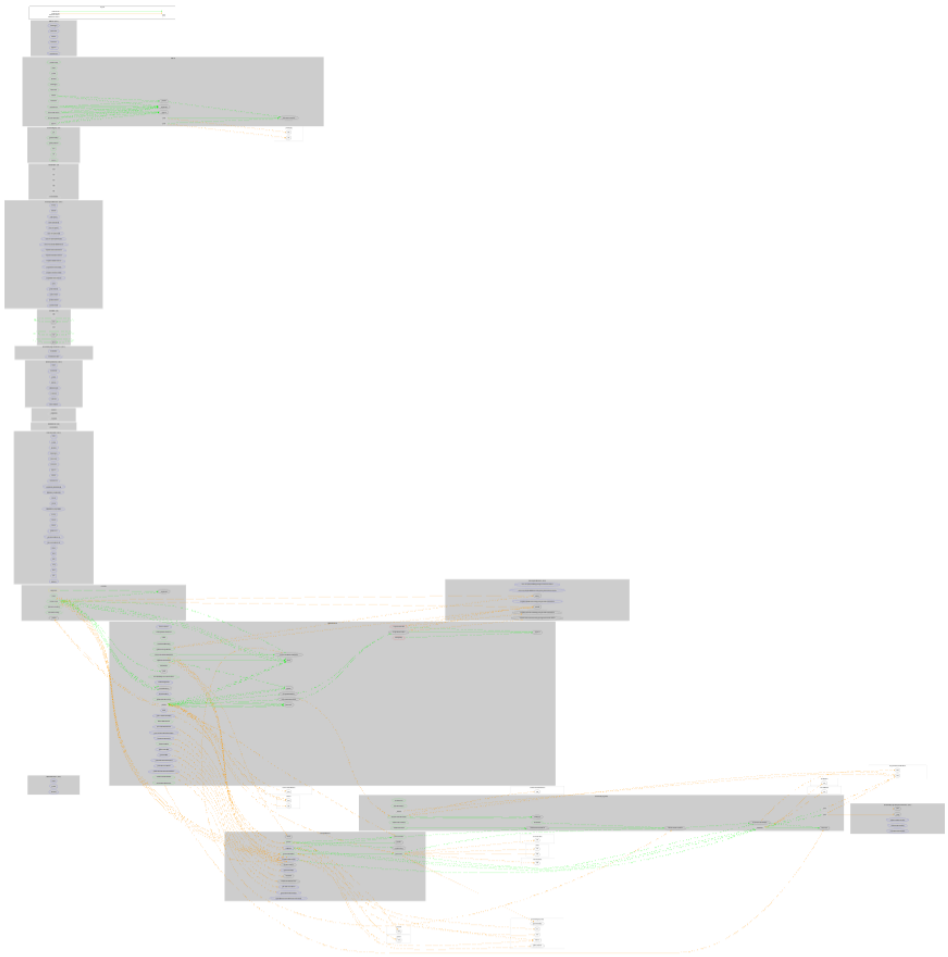
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	
Unverified / Not checked	✗
Not available	-

CallGraph



Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/IERC20Metadata.sol	———	1	27	16	4	15	9	🌟
📝	contracts/DividendPayingToken.sol	1	———	177	177	90	51	82	🔺
🔍	contracts/IUniswapV2Pair.sol	———	1	54	9	5	1	55	———
📚	contracts/SafeMathUint.sol	1	———	15	15	8	5	3	———
🎨	contracts/Context.sol	1	———	24	24	10	12	1	———
🔍	contracts/IUniswapV2Factory.sol	———	1	19	8	4	1	17	———
🔍	contracts/DividendPayingTokenInterface.sol	———	1	36	13	3	16	5	———
📚	contracts/SafeMath.sol	1	———	146	146	39	93	10	🌟
🔍	contracts/IUniswapV2Router.sol	———	2	142	7	4	2	64	💰
📚	contracts/SafeMathInt.sol	1	———	92	92	33	47	16	———
📝	contracts/Ownable.sol	1	———	57	57	27	21	25	———
📚	contracts/IterableMapping.sol	1	———	63	63	49	2	19	———
📝	contracts/hero_battle.sol	1	———	440	418	289	22	294	💰🔺🌀
📝	contracts/ERC20.sol	1	———	310	294	85	178	82	———
📝	contracts/DividendTracker.sol	1	———	221	203	144	2	95	———
🔍	contracts/DividendPayingTokenOptionalInterface.sol	———	1	25	13	3	14	7	———
🔍	contracts/IERC20.sol	———	1	81	26	17	57	13	🌟
📝📚🔍🎨	**Totals**	**10**	**8**	**1929**	**1581**	**814**	**539**	**797**	💰🔺🌀🌟

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	hero_battle.sol	Missing Zero Address Validation (missing-zero-check)	184, 134, 161, 56	Check that the address is not zero.
#2	hero_battle.sol	A floating pragma is set	2	The current pragma Solidity directive is ""^0.6.2"".
#3	dividendtracker.sol	A floating pragma is set	2	The current pragma Solidity directive is ""^0.6.2"".
#4	dividendtracker.sol	Missing Zero Address Validation (missing-zero-check)	43, 74, 146, 210	Check that the address is not zero.

Informational issues

Issue	File	Type	Line	Description
#1	hero_battle.sol	State variables that could be declared constant (constable-states)	23	Add the `constant` attributes to state variables that never change.

Audit Comments

18. August 2021:

- There is still an owner (Owner still has not renounced ownership)

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

AUDITEK

BLOCKCHAIN SOLUTIONS AND CONSULTING