Wixpool Rate Stabilizer Code Review

VERSION 1: *10/11/2022*

VERSION 2: *17/01/2023*

Audited by – AHR (OF-10) The_Swarm

THIS ASSESSMENT REPORT HAS BEEN TESTED, COMPLETED AND AUTHORISED ON 24/04/2019
APPROVED BY THE CEO | BLUESWARM LTD COMPANY NUMBER 11489054
BlueSwarm

wixpool & BlueSwarm

Executive Summary

The purpose of this report

Smart Contract Code Review

This document aims to record the vulnerabilities found from a code review conducted by Blueswarm. The detected vulnerabilities are plotted against Best Practice Guidelines laid down by the community.

The Objective

Blueswarm to perform an industry best practice Vulnerability Assessment and Code Review and reports the findings from the following smart contracts only:

WRS Algorithm, Wixpool Gate Defence

Execution Strategy

Our execution strategy incorporates proven methodologies, extremely qualified personnel, and a highly responsive approach to managing deliverables and the utilization of proprietary software.

Methodology

The code audit was carried out using the specification of SWC (Smart Contract Weakness Classification) and CWE (Common Weakness Enumeration). The assessment was conducted using a combination of proprietary software and manual testing by highly skilled individuals.

Vulnerability Overview

Timeline and Audit Log

The Security Code Audit for the **WRS Algorithm** of **Wixpool project** lasted 23 days from the 17th Sep 2022 to 10th Oct June 2022. Where in total, 2 contracts: *WRS Algorithm, Wixpool Gate Defence*

Vulnerabilities Detected

A total of 10 vulnerabilities were discovered and identified in the contracts.
The following and below mentioned charts show the respective severity classifications used, the breakdown and distribution of vulnerabilities.

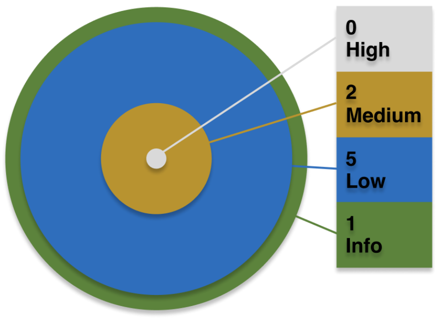


Fig 2. Vulnerability Breakdown in Numbers

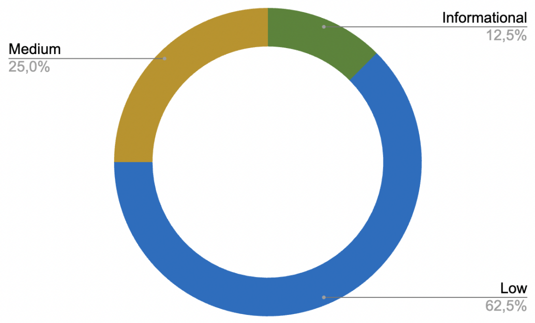


Fig 3. Vulnerability Distribution in %

HIGH RISK – No problems of high severity were found

WRS Algorithm and Wixpool Gate Defence are working properly

No threat to user assets

MEDIUM RISK - A total of 2 classified as medium risk vulnerabilities detected

Contract Files Affected – *Wixpool Gate Defence*

LOW RISK - A total of 5 classified as low risk vulnerabilities detected Contract

Files Affected: *WRS Algorithm, Wixpool Gate Defence*

Exploit Effort & Resource Classification

Rating	Definition of Risk Rating	Definition of Resource Requirement to Exploit	Definition of Effort to Exploit
HIGH	Deficiency creates a vulnerability that could result in loss of system control or override a desired function or give access to critical or sensitive information.	Recommendation either requires the purchase of hardware or, requires significant research and resources to exploit	To exploit the weakness requires a high level of expertise and advanced knowledge of smart contract design, and programming
MEDIUM	Deficiency creates an exposure to a larger, but limited loss of confidentiality or integrity, as the result of many user accounts being compromised, or restricted functions being accessed.	Recommendation may require the purchase of hardware or software and/or requires moderate, research and implementation activities to exploit	Requires medium level of effort. No tools are available but sample code or other similar exploits are known
LOW	Deficiency creates limited exposure to the compromise of user accounts or unauthorized access to data	Recommendation may require the purchase of minor hardware or software and/or requires minor research and implementation activities to exploit	Easy to exploit with known methods or tools with minimal modifications

Exploit Efforts & Resource Analysis

The following graphs below provide insight into the exploit efforts and resources needed in order to successfully complete or carry out exploitation mapped against the 10 vulnerabilities detected

Exploit Effort

Of the 10 Security issues currently identified, all vulnerabilities would require a low level of to exploit

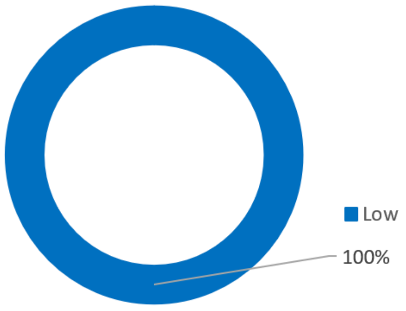


Fig 4. Exploit Effort Breakdown in %

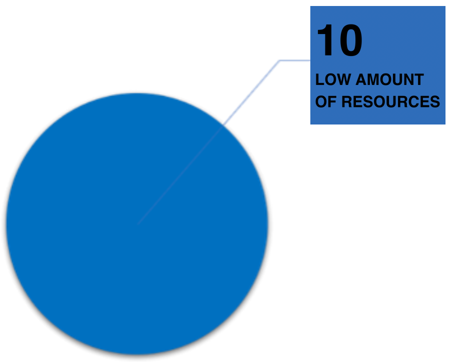


Fig 5. Exploit Effort Breakdown in numbers

Exploit Resource Requirements

Of the 10 Security issues identified, all vulnerabilities that can be exploited require less resources to exploit.

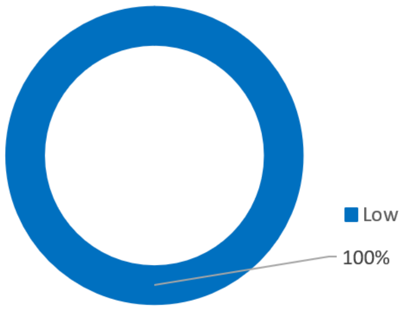


Fig 6. Resources need to exploit in %

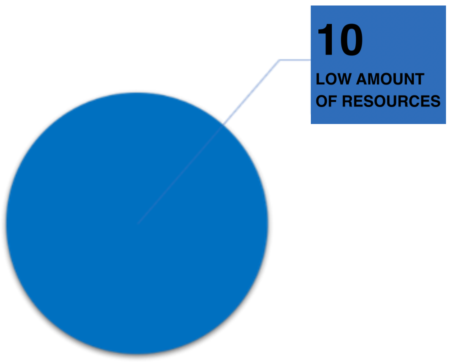


Fig 7. Resources needed to exploit in numbers

Remediation Resource Requirements

Rating	Definition of Risk Rating	Definition of Resource Requirement to Remediate	Definition of Effort to Remediate
HIGH	Deficiency creates a vulnerability that could result in loss of system control or override a desired function or give access to critical or sensitive information.	Recommendation either requires the purchase of hardware or, requires significant changes to the code base or research and resources to remediate	To remediate the vulnerabilities requires a high level of expertise and advanced knowledge of smart contract design, and programming
MEDIUM	Deficiency creates an exposure to a larger, but limited loss of confidentiality or integrity, as the result of many user accounts being compromised, or restricted functions being accessed.	Recommendation may require the purchase of hardware or software and/or requires moderate changes to the codebase and/or research and implementation activities to remediate the vulnerability	Requires medium level of effort and changes to remediate.
LOW	Deficiency creates limited exposure to the compromise of user accounts or unauthorized access to data	Recommendation may require the purchase of minor hardware or software and/or requires minor changes in the codebase to remediate against the vulnerability	Easy to remediate with minimal modification or effort

Remediation Resource Requirements

Of the 10 Security issues identified, remediation efforts and resources required in all circumstances are considered Low. Therefore, minimal resources and programming efforts are required to implement satisfactory remediation.

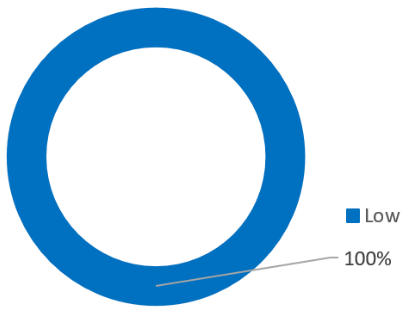


Fig 8. Resources need to remediate in %

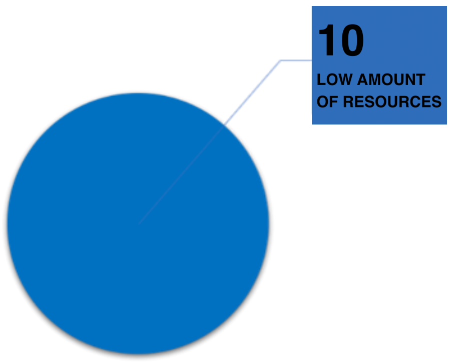


Fig 9. Resources needed to remediate in numbers

Severity

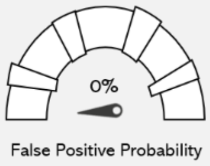
MEDIUM

Contract Name/s

Category: . Violation of Check-Effects

List of Contracts Affected

Wixpool Gate Defence



100%
True Positive Probability

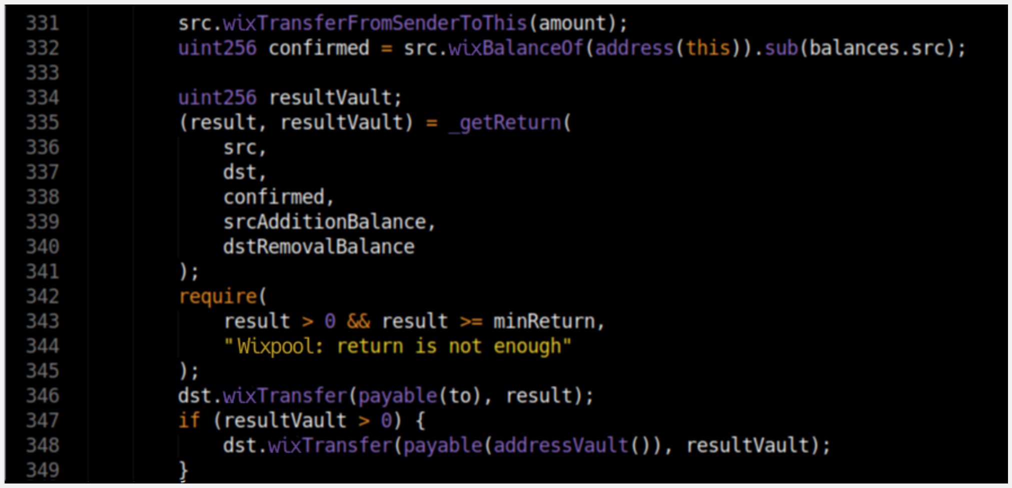
STATUS
CLOSED

Description

State Variables updated after External Calls . Violation of Check-Effects Interaction Pattern. The Wixpool Gate Defence contract includes the swap function that updates some of the very imperative state variables of the contract after the external calls are being made. An external call within a function technically shifts the control flow of the contract to another contract for a particular period of time. Therefore, as per the Solidity Guidelines, any modification of the state variables in the base contract must be performed before executing the external call. Although the function has been assigned the nonReentrant modifier, the approach used, in this function, for making an external call violates the

Check Effects Interaction Pattern. The following functions in the contract updates the state variables after making an external call at the lines mentioned below: ● swap() function at Line 331, 346 and 348

Code Reference/s Line 331, 346 and 348



```
331        src.wixTransferFromSenderToThis(amount);
332        uint256 confirmed = src.wixBalanceOf(address(this)).sub(balances.src);
333
334        uint256 resultVault;
335        (result, resultVault) = _getReturn(
336            src,
337            dst,
338            confirmed,
339            srcAdditionBalance,
340            dstRemovalBalance
341        );
342        require(
343            result > 0 && result >= minReturn,
344            "Wixpool: return is not enough"
345        );
346        dst.wixTransfer(payable(to), result);
347        if (resultVault > 0) {
348            dst.wixTransfer(payable(addressVault()), resultVault);
349        }
```

Severity

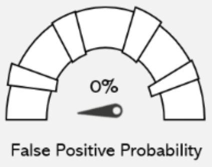
MEDIUM

Category: Violation of Check_Effects Interaction Pattern found in the contract

Contract Name/s

List of Contracts Affected

Wixpool Gate Defence



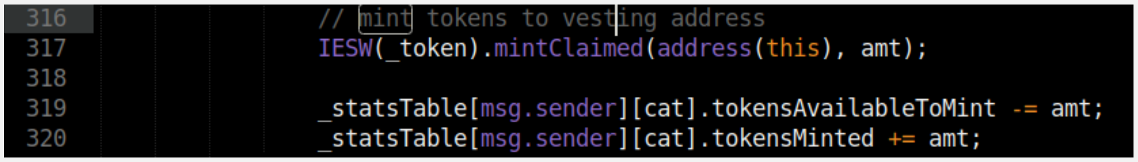
STATUS CLOSED

Description

As per the Check_Effects_Interaction Pattern in Solidity, external calls should be made at the very end of the function and event emission, as well as any state variable modification, must be done before the external call is made. The following functions, however, violate the Check-Effects Interaction pattern:

Code Reference/s

_burnLock at Line 363-365 ● mint() at Line 317-320



```
316            // mint tokens to vesting address
317            IESW(_token).mintClaimed(address(this), amt);
318
319            _statsTable[msg.sender][cat].tokensAvailableToMint -= amt;
320            _statsTable[msg.sender][cat].tokensMinted += amt;
```

Remediation

Check Effects Interaction Pattern must be followed while implementing external calls in a function.

Severity

LOW

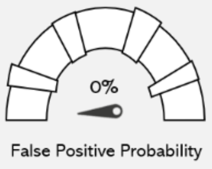
Contract Name/s

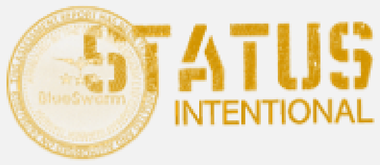
Category: State Variable initialized but.

never used in the contract

List of Contracts Affected

Wixpool Gate Defence
WRS Algorithm





Description

Explanation: The Crowdsale contract includes a state variable defRef, with an internal visibility, that is being initialized but never used throughout the gas. Recommendation: State variables should either be used effectively in the contract or removed to reduce gas usage.

Code Reference/s

Line no - 56

adress internal defRef

Remediation

State variables should either be used effectively in the contract or removed to reduce gas usage

Severity

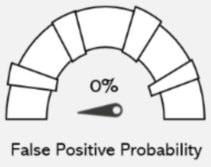
LOW

Contract Name/s

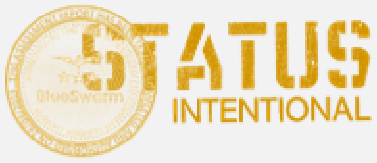
Category: presaleBulkLoad function includes a Hardcoded date

List of Contracts Affected

Wixpool Gate Defence
WRS Algorithm



100%
True Positive Probability

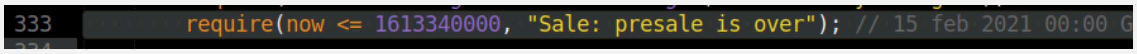


Description

Keeping in mind the immutable nature of smart contracts, it is not considered a better practise to hardcode any address or imperative uint in the contract before deployment.

Code Reference/s

Line: 333



```
333        require(now <= 1613340000, "Sale: presale is over"); // 15 feb 2021 00:00 G
```

Remediation

It is recommended to use a state variable for storing such an integer and initialize them in the constructor.

Severity

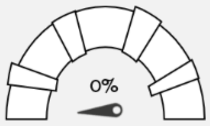
LOW

Contract Name/s

Category: Boolean Constant is being inadequately used in the buyWithETH function

List of Contracts Affected

WRS Algorithm



False Positive Probability

True Positive Probability

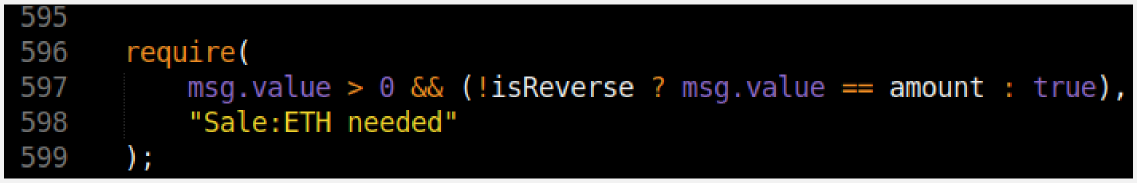
STATUS
INTENTIONAL

Description

During the automated testing of the crowdsale contract, it was found that the buyWithEth function includes a require statement that doesn't implement proper usage of boolean constant.

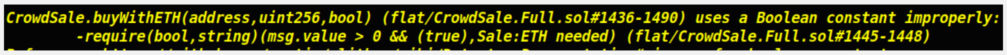
Code Reference/s

Line: - 596-599



```
595
596    require(
597        msg.value > 0 && (!isReverse ? msg.value == amount : true),
598        "Sale:ETH needed"
599    );
```

Automated Test Result



```
CrowdSale.buyWithETH(address,uint256,bool) (flat/CrowdSale.Full.sol#1436-1490) uses a Boolean constant improperly:
        -require(bool,string)(msg.value > 0 && (true),Sale:ETH needed) (flat/CrowdSale.Full.sol#1445-1448)
```

Remediation

It is recommended to modify the require statement and implement the boolean constant usage correctly.

Severity

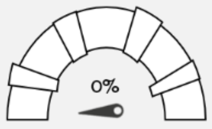
LOW

Category: too many digits used

Contract Name/s

List of Contracts Affected

Wixpool Gate Defence



False Positive Probability

True Positive Probability

STATUS

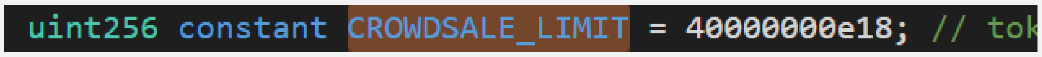
CLOSED

Description

The mentioned lines have a large number of digits that makes it difficult to review and reduce the readability of the code. The following State Variables/Functions of respective contracts mentioned below include large digits:

Code Reference/s

a. Wixpool Gate Defence ● CROWDSALE_LIMIT at Line 22



```
uint256 constant CROWDSALE_LIMIT = 40000000e18; // tok
```

b. Wixpool Gate Defence

quorumVotes() 21-23

proposalThreshold() 26-28

Remediation

Ether Suffix could be used to symbolize the 10^18 zeros

Severity

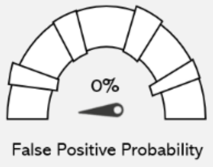
LOW

Category: External Visibility should be preferred

Contract Name/s

List of Contracts ffected

Wixpool Gate Defence
WRS Algorithm



100%
True Positive Probability

STATUS CONSIDERED

Description

Those functions that are never called throughout the contract should be marked as external visibility instead of public visibility. This will effectively result in Gas Optimization as well. Therefore, the following functions of respective contracts mentioned below should must be marked as external within the contract:

Code Reference/s

WRS ● updateParams ● stopCrowdSale ● setPoolsize ● fetchCoin ● setStatusByID ● setRateByID ● coinCounter() ● coin() coinRate() ● coinData() ● presaleBulkLoad ● buy()

Wixpool Gate Defence ● getPoolDataList() ● getReservesByPool() ● ● getReserves() getExpectedReturn() ● removeLiquidity() ● removeLiquidityETH()

WRS ● propose ● queue () ● execute () ● cancel () ● getReceipt () ● castVote () ● castVoteBySig() ● __acceptAdmin() ● __abdicate() ● __queueSetTimelockPendingAdmin() ● __executeSetTimelockPendingAdmin()

Remediation

If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

Severity

INFORMATIONAL

Contract Name/s

Category: Coding Style Issues in the contract

List of Contracts affected

Wixpool Gate Defence
WRS Algorithm

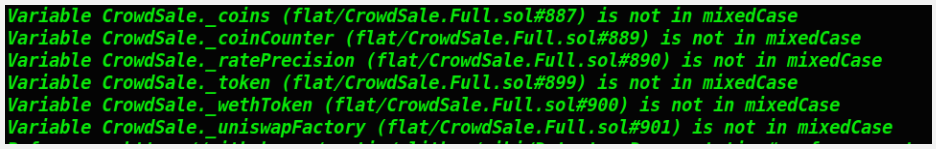
BlueSwarm

STATUS
NOT CONSIDERED

Description

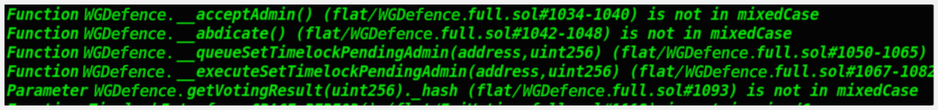
Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future. During the automated testing, it was found that the following contracts have quite a few code style issues:

a. CrowdSale



```
Variable CrowdSale._coins (flat/CrowdSale.Full.sol#887) is not in mixedCase
Variable CrowdSale._coinCounter (flat/CrowdSale.Full.sol#889) is not in mixedCase
Variable CrowdSale._ratePrecision (flat/CrowdSale.Full.sol#890) is not in mixedCase
Variable CrowdSale._token (flat/CrowdSale.Full.sol#899) is not in mixedCase
Variable CrowdSale._wethToken (flat/CrowdSale.Full.sol#900) is not in mixedCase
Variable CrowdSale._uniswapFactory (flat/CrowdSale.Full.sol#901) is not in mixedCase
```

b. Wixpool Gate Defence



```
Function WGDefence.__acceptAdmin() (flat/WGDefence.full.sol#1034-1040) is not in mixedCase
Function WGDefence.__abdicate() (flat/WGDefence.full.sol#1042-1048) is not in mixedCase
Function WGDefence.__queueSetTimelockPendingAdmin(address,uint256) (flat/WGDefence.full.sol#1050-1065)
Function WGDefence.__executeSetTimelockPendingAdmin(address,uint256) (flat/WGDefence.full.sol#1067-1082
Parameter WGDefence.getVotingResult(uint256)._hash (flat/WGDefence.full.sol#1093) is not in mixedCase
```

Remediation

Therefore, it is highly recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract

Open Cases

Issues	Open Issues	Closed Issues
Critical Severity		
Medium Severity		2
Low Severity		5
Information	1	
Total Found	1	9

Conclusion

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review etc. All the issues have been explained and discussed in detail above. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned. Out of the vulnerabilities found, all vulnerabilities and issues identified were either corrected or had been adequately addressed through other controls.
either the contract in question have been deprecated or future low effect fixes will be modified in future upgrades.

DISCLAIMER [CLIENT: WIXPOOL

V1: Original Report without remediation [ORIGINALTESTDATE]: 10/11/2022

V2: Remediation Report [REMEDIATIONTESTDATE]: 17/01/2023

This review is marked as V.2, which was conducted by Blueswarm's certified security engineers. We identified several security vulnerabilities and provided remediation advice to Wixpool

After being notified by [CLIENT] that all vulnerabilities have been corrected, Blueswarm have performed a remediation test (V.2) on [REMEDIATIONTESTDATE] to confirm that all vulnerabilities and issues identified were either corrected or had been adequately addressed through other controls. While no application or system can be 100% secure, all of our security findings were corrected or addressed and it is our opinion that the contracts tested are reasonably well written from a security perspective and the applications and supporting systems are deployed, configured and implemented in a secure manner. IF NOT FULLY CORRECTED The review was conducted by Blueswarms's certified security engineers. We identified several security vulnerabilities and provided remediation advice to [CLIENT]. After being notified by [CLIENT] that these selected vulnerabilities had been corrected, Blueswarm performed a remediation test on
[REMEDIATIONTESTDATE] and confirmed that these selected vulnerabilities were either corrected or had been adequately addressed through other controls. There were findings identified by Blueswarm that were not validated as corrected. Please contact [CLIENT] for further information regarding these findings and their resolution status. DISCLAIMER: Blueswarm conducted this testing on the smart contracts that existed as of [ORIGINALTESTDATE]. Information security threats are continually changing, with new vulnerabilities discovered on a daily basis, and no application can ever be 100% secure no matter how much security testing is conducted. This report is intended only to provide documentation that [CLIENT] has corrected all findings noted by Blueswarm as of [REMEDIATIONTESTDATE]. This report cannot and does not protect against personal or business loss as the result of use of the applications or systems described. Blueswarm offers no warranties, representations or legal certifications concerning the applications, code or systems it tests. All software includes defects: nothing in this document is intended to represent or warrant that security testing was complete and without error, nor does this document represent or warrant that the application tested is suitable to task, free of other defects than reported, fully DISCLAIMER - Compliant with any industry standards, or fully compatible with any operating system, hardware, or other application. By using this information you agree that Blueswarm shall be held harmless in any event