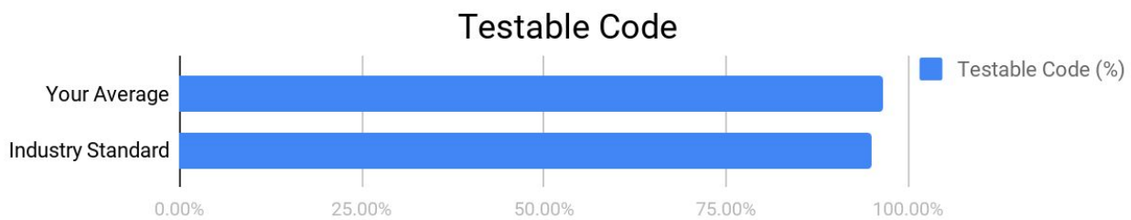
Intimate Contract Audit

by Hosho, April 2018

Executive Summary

This document outlines the overall security of Intimate smart contract as evaluated by Hosho's Smart Contract auditing team. The scope of this audit was to analyze and document Intimate's token contract codebase for quality, security, and correctness.

All issues have been remediated. (See Complete Analysis)



Testable code is higher than industry standard. (See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Hosho recommend that the Intimate Team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table Of Contents

1. Auditing Strategy and Techniques Applied

The Hosho Team has performed a thorough review of the smart contract code, the latest version as written and updated on April 11, 2018. All main contract files were reviewed using the following tools and processes. (See All Files Covered)

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing ERC-20 Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste; and
- Uses methods safe from reentrance attacks.
- Is not affected by the latest vulnerabilities

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of Intimate's token contract. To do so, it has been reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they were discovered. Part of this work included writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.
5. Deploying the smart contract to testnet and production networks using multiple client implementations to run live tests.

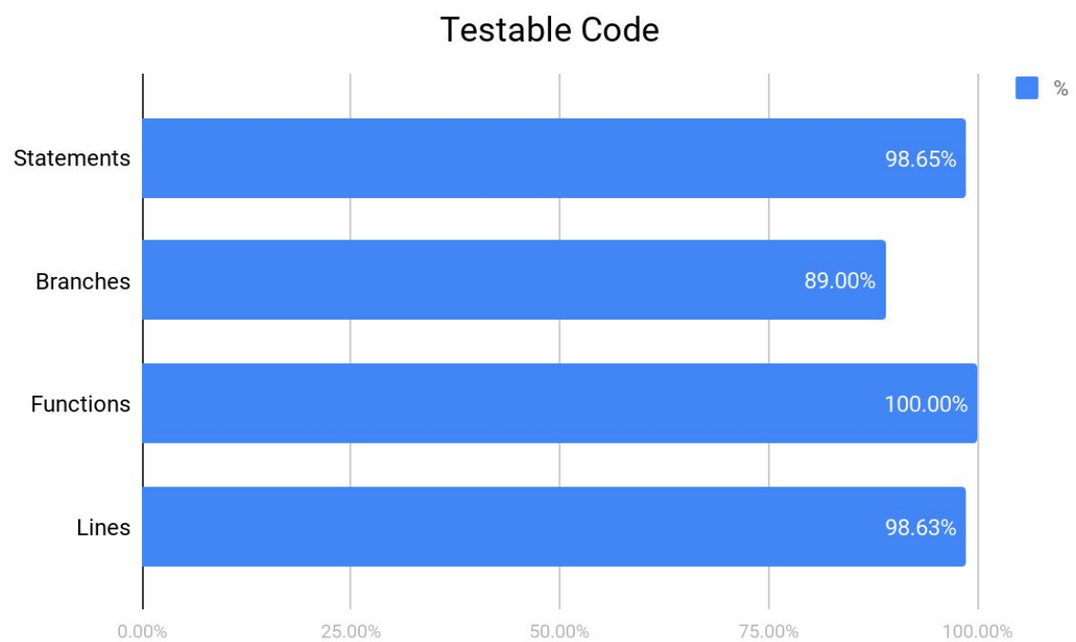
2. Structure Analysis and Test Results

2.1. Summary

The Intimate contracts comprise a standard ERC-20 compliant token with associated crowdsale.

2.2 Coverage Report

As part of our work assisting Intimate in verifying the correctness of their contract code, our team was responsible for writing a unit test suite using the Truffle testing framework.



For individual files see Additional Coverage Report

2.3 Failing Tests

No failing tests.

See Test Suite Results for all tests.

3. Complete Analysis

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Informational** - The issue has no impact on the contract’s ability to operate.
- **Low** - The issue has minimal impact on the contract’s ability to operate.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

1.1. Resolved, Critical: Token Removal

RecoverCurrency

Explanation

The vesting system for ITM does not limit the addresses that can be used as approved addresses for the `vestingFunds` function, which holds the tokens designated for staking. Due to this, the ERC-20 contract could be named the holder of funds for vesting. This would allow the administrators to use the recovery function to remove tokens that are reserved for the vesting system.

Resolution

The vesting code has been updated to a dedicated vesting contract from OpenZeppelin, solving this issue.

1.2. Resolved, Critical: Private Key Visibility

VestingToken

Explanation

The private keys for the ETH addresses, which are provided as entries to the staking system, are delivered through a Ganache static mnemonic which allows for recovery of these keys. As the Hosho final audit contains a checksum of the files contained in the repository, any changes by Intimate to these pre-written vesting schedules, for deployment or testing purposes, will invalidate the checksum for this file.

Resolution

The vesting code has been updated to a dedicated vesting contract from OpenZeppelin, solving this issue.

1.3. Resolved, Critical: Array Order Bug

VestingToken

Explanation

During the removal of addresses from the vesting list via the `revoke` function, the vesting schedule locations are not updated accordingly. Given that, it is possible that the remaining addresses will be stored as keys in a continuous location that can be altered to 0 that later functions use as deleted and then revert, altering the intended function of the contract as a whole.

If these two structures are to remain consistent, they both need to be updated in a consistent manner. As it stands, both of these structures are updated in different manners, which leads to inconsistent results when calling the address from either the `vestingSchedule` or `vestingRecord`.

Additionally, if a new schedule is pushed, it will use the length of the `vestingRecords` array, and assign it to a potentially unwanted address.

Note: The vesting system itself is not affected, as it relies on the array length, rather than any external mapping.

Technical Example

Assume there are 4 entries listed as part of the `vestingSchedule` map. If the first 2 are removed, indexes 3 and 4 in the `vestingSchedule` will be shifted to array entries 0 and 1, based on how the contract is interacting with these structs. As these correspond to the `vestingRecords` entries 2 and 3, which can no longer be accessed, it will attempt to take index 3 or 4, subtract one, then revert which is not intended behavior for valid address entries.

Resolution

The vesting code has been updated to a dedicated vesting contract from OpenZeppelin, solving this issue.

1.4. Resolved, Critical: No Address Validation

BasicToken

Explanation

The `bulkTransfer` function does not validate address cases, which creates the following issues:

1. Unintended destruction of tokens
2. Potentially enabling over-transfers from mis-entered addresses

For the first case, msg.sender is set to 0 during initialization in order to prevent a re-entrancy attack which is common best practice. However, there is no check to prevent msg.sender from being added to the approved `bulkTransfer` list of addresses. As any remaining tokens are assigned, they are not added to the address intended. This will overwrite any balance on the address, including tokens sent during the `bulkTransfer` process, which results in the unintentional destruction of the tokens.

The second case is caused by the lack of validation for checking if an address has already been sent to, which means that an address could be entered twice into the array for transfer.

Resolution

The code has been updated to disallow msg.sender from being added to the approved list of addresses, preventing both of these cases from happening.

1.5. Resolved, High: No Array Length Validation

BasicToken

Explanation

The `bulkTransfer` function takes two arrays, one of addresses and one of values. However, these directly related arrays are not compared to verify that both contain an equal number of entries. Due to this, if the length of the `_values` array is smaller than the `_tos` length, it will cause an invalid array access, and revert.

Resolution

Validation has been added to ensure these arrays are equal in length.

1.6. Resolved, High: Infinite Length Array

BasicTokenStorage

Explanation

The `trackAddresses` functionality is utilized to track all addresses seen by the contract. It does this by keeping track of an internal array, named `accounts`, and an internal mapping, named `seenBefore`. Both of these are added to every time a new address is observed, but never removed from the system, even if balances are dropped to 0. This causes the array length to grow without a limit as it is unrestricted.

Resolution

A function named `removeSeenAddress` has been added to account for this case.

1.7. Resolved, High: Incorrect minValue

IntimateShoppe

Explanation

The rate of ITM per WEI can be set during initialization while the `minValue` is a constant equal to 1 ETH divided by 600. If the `minValue` is not reset based on the rate determined during initialization, the 1 ITM minimum worth of ETH becomes not valid.

Resolution

The rate is now being used to set the `minVal`, ensuring it is the correct value.

1.8. Resolved, Medium: Incorrect Event Issuance

RecoverCurrency

Explanation

The contract is emitting a Transfer event for a non-token operation, the recovery of Ethereum, which will cause an incorrect value track on 3rd party sites.

Resolution

The Transfer event has been updated to a non-Transfer event.

1.9. Resolved, Medium: Event Issuance Utilized as System Error

BasicToken

Explanation

The EVM allows for fully atomic operations to be executed via the revert() system. There are three separate cases written into the `bulkTransfer` function that utilize events instead of the existing system. If each of these cases was handled as atomic operations, they would fail when executed as opposed to issuing events which leads to inaccurate logging, particularly for third parties, as well as not properly handling the previous transactions that the revert handles.

Those cases are as follows:

1. 0x0 check, which skips, and does not create a notification
2. The insufficient funds check, which allows a partially completed bulk transfer, making the only possible way to validate full completion is to check for the lack of an event
3. The overflow check, which would revert anyway due to the utilization of SafeMath, but issues an event regardless.

Resolution

Each of these cases has been removed and the revert() system is allowed to operate as intended.

1.10. Resolved, Medium: Token Decimal Change

TokenSettings

Explanation

The decimal system in the ERC-20 contract is used to denote what constitutes a "full" token. Generally, this is defined as 1e18 atomic units or 18 digits after the decimal. Any changes to this value will alter the number of "full" tokens being transferred. Without creating an additional contract, there is no way to guarantee that the crowdsale is completed in the same atomic transaction. Since the token issuance in the Shoppe contract is completed in atomic units, there exists the possibility for an incorrect sale amount to be issued if the value in the ERC-20 contract is decreased.

Resolution

The `setDecimals` function in this contract has been removed by the Intimate team.

1.11. Resolved, Low: Invalid Initialization

VestingToken

Explanation

All token transfers within the Intimate ecosystem utilize the `trackAddresses` feature to log the owners of ITM tokens. The initialization function for `VestingToken`, which is capable of issuing tokens, does not utilize this system.

Resolution

The vesting code has been updated to a dedicated vesting contract from OpenZeppelin, solving this issue.

1.12. Resolved, Low: Invalid Initialization

Aphrodite

Explanation

All token transfers within the Intimate ecosystem utilize the `trackAddresses` feature to log the owners of ITM tokens. The initialization function for Aphrodite, which is capable of issuing tokens, does not utilize this system.

Resolution

The tracking function used in the rest of the contracts has been added to this contract.

---

4. Closing Statement

We are grateful to have been given the opportunity to work with the Intimate Team.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Intimate contract is free of any critical issues.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

We at Hosho recommend that the Intimate Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Coverage Report:

Contract: ERC-20 Tests for Aphrodite

✓ Should deploy a token with the proper configuration (83ms)

✓ Should allocate tokens per the minting function, and validate balances (394ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (142ms)

✓ Should not transfer negative token amounts (38ms)

✓ Should not transfer more tokens than you have

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (62ms)

✓ Should not allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer an additional 500 tokens once authorized, and authorization balance is > 0

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (64ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (204ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (41ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0 (46ms)

✓ Should not transfer tokens to 0x0

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

Contract: Extended token tests for Intimate/Aphrodite

Aphrodite

✓ Should accept ETH sent to it, and issue an event (92ms)

BasicToken

✓ Should not allow for self-transfer (84ms)

Bulk Transfers

✓ Should block transfers to 0x0, reverting if one is hit, to maintain state (87ms)

✓ Should revert if there is not a high enough balance on the source address (84ms)

✓ Should revert on any transfers back to the source address

✓ Should revert on non-balanced array lengths (56ms)

✓ Should transfer tokens, issuing a Transfer event for each transfer done (206ms)

RecoverCurrency

✓ Should allow token recovery (308ms)

✓ Should not allow a 0x0 address to be passed to recoverToken (73ms)

✓ Should allow the recovery of ETH, and not issue a Transfer event (115ms)

TokenSettings

✓ Should allow the name to be changed (42ms)

✓ Should allow for the symbol to be changed (48ms)

TokenLedger

✓ Should return back the number of addresses that have a balance after initialization

✓ Should return the account array

✓ Should return the balance of the account corresponding to the address at the given array index

✓ Should revert if the ID is outside of the address array length

Pausable

✓ Should be able to be paused, if not in a paused state (123ms)

✓ Should be able to be unpaused, if not in an unpaused state (198ms)

Authorized

✓ Should not allow a user to change their own authorization

✓ Should return back authorization states (48ms)

✓ Should allow setting/revocations of authorizations, and overriding of CUPID with APHRODITE (181ms)

Contract: IntimateShoppe

Configuration

- ✓ Should require valid deployment parameters (287ms)
- ✓ Should allow the round to be set at any time (264ms)
- ✓ Should only allow the maxValue to be set when the sale is not running (450ms)
- ✓ Should only allow the setMinValue to be set when the sale is not running (436ms)
- ✓ Should only allow round times to be changed if the sale isn't running (391ms)
- ✓ Should only allow the token cap to be set if the sale isn't running (466ms)
- ✓ Should only allow the sale rate to be set if the sale isn't running (480ms)
- ✓ Should allow the company wallet to be changed (94ms)
- ✓ Should let the high water line be pulled, and changed (48ms)
- ✓ Should return the contributors to the contract
- ✓ Should return the contributions for an address

Token Purchases

- ✓ Should require that you submit more than the min value of eth
- ✓ Should require that you don't submit more than the max value of eth
- ✓ Should require that the sale is started
- ✓ Should not allow the purchase of enough tokens to go over the cap (193ms)
- ✓ Should properly handle sales and fund transfers when highwater is exceeded (719ms)

6. All Contract Files Tested

File	Fingerprint (SHA256)
auth/Authorized.sol	e435ba0907e3b61a0693b586f6a3c2c00d65213d57ad266a58e1da45a83e467b
auth/AuthorizedList.sol	e0b512d6354332e944ed38980b95b7053b11c1e2810d2fc82b1efc26ed3c694a
managed/Freezable.sol	cf678b910b711ca47b5bcef5259ec3b992190d8724f43b1feba74bdcd96eafc3
managed/Pausable.sol	29c464a300665a15f0961f5159b209ce9765f9b51ce3eccd5a293178d6ef46e1
math/SafeMath.sol	38fac60b01f7d76dbb9beb8d2ca3ab1dad1ae2ea28bf0882c1924e29708d2fb3
sales/IntimateShoppe.sol	202ce6206e278594285389b6035031bf1047b8b515d1b11290bac4c8cfc7beb1
storage/AllowancesLedger.sol	f484d59c5d049aa9ed376aae22d73c8d6a073068061b37d4d228ede2c96c22a3
storage/BasicTokenStorage.sol	1fb59c9eb107b64c53c3499aef9f2b4cd9818161e19a310974307109bddd3d15
storage/TokenLedger.sol	07020457d1c639c434ac862ab0383a999464b75e57d950a8640db826dfa3fe35
storage/TokenSettings.sol	d219195c921f03183f883eecadd437c2e06746d93f6a294cd053e45cbb2c948c
token/Aphrodite.sol	257a40dfcd3d5d7d55c04f2234b37d993d29f73e46193c02061c8c67e5901399
token/BasicToken.sol	19c38b806c1d0c7b12b8c866d09d20f65c4e3cecf517f6f83be1915ac7096e5a
token/IERC20.sol	63595144db29c8b1c7c8800b3d2f5a69784f9f74726763973db4f00a3e5b5082
token/IERC20Basic.sol	0903847cd535a36419ed570dc754d450862b15f96fc4f0b779605a112b4a97a9
token/RecoverCurrency.sol	7a3310c5c2c55676328ec5a5d494e55c14dc361380209365ddba1878a8da3df6

7. Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
auth/Authorized.sol	100.00%	100.00%	100.00%	100.00%
auth/AuthorizedList.sol	100.00%	100.00%	100.00%	100.00%
managed/Freezable.sol	100.00%	100.00%	100.00%	100.00%
managed/Pausable.sol	100.00%	100.00%	100.00%	100.00%
math/SafeMath.sol	100.00%	66.67%	100.00%	100.00%
sales/IntimateShoppe.sol	100.00%	97.37%	100.00%	100.00%
storage/AllowancesLedger.sol	100.00%	100.00%	100.00%	100.00%
storage/BasicTokenStorage.sol	85.71%	66.67%	100.00%	900.00%
storage/TokenLedger.sol	100.00%	100.00%	100.00%	100.00%
storage/TokenSettings.sol	100.00%	100.00%	100.00%	100.00%
token/Aphrodite.sol	100.00%	100.00%	100.00%	100.00%
token/BasicToken.sol	96.67%	88.89%	100.00%	96.30%
token/IERC20.sol	100.00%	100.00%	100.00%	100.00%
token/IERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
token/RecoverCurrency.sol	100.00%	100.00%	100.00%	100.00%
token/StandardToken.sol	100.00%	66.67%	100.00%	100.00%
All files	**98.65%**	**89.00%**	**100.00%**	**98.63%**