Gladius Token Contract Audit

by Hosho, October 2017

Table of Contents

Technical Summary

This document outlines the overall security of Gladius' smart contract as evaluated by Hosho's Smart Contract auditing team.

The scope of this audit was to analyze and document Gladius' token contract codebase for quality, security, and correctness.

The quality of the contracts are high and has been found free of security issues. There was a small decrease in coverage in the re-audit due to new, un-used code paths, but these have been scanned for validity. Very few changes were required in Hosho's tests which is representative of good coverage.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, merely an assessment of its 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 Gladius Team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Auditing Strategy and Techniques Applied

The Hosho Team has performed a thorough review of the smart contract code as written and last updated on October 8, 2017. The following contract files and their respective SHA256 fingerprints were evaluated:

File	Fingerprint (SHA256)
infrastructure/ITokenRetreiver.sol	d40b4dd6f82175253149bb2905c8addb6bb21bc3850f16a6333f43cc61bcd4e3
infrastructure/authentication/whitelist/IWhitelist.sol	47c9c59c9868ac708964548ce17b9ff963a42550e7fd66e7cae47bf0e1b90450
infrastructure/authentication/whitelist/Whitelist.sol	1481959fe5c29a091d1212a1e246594743ebfbe1dbb7fb5811e88e8d7254966f
infrastructure/modifier/InputValidator.sol	cc06d78630510d66cefa6a537e8cf27e1756f66058876a201417ab983daf1ec7
infrastructure/modifier/Owned.sol	0054fb56c086e3dc1376c8fa9b8521c21fdad1494692fb681b19193a454c6da2
infrastructure/ownership/IOwnership.sol	baeb0fe923b7e8c0e53d23339e1b06d49c9581a854e342eef710d0d7cdea5892
infrastructure/ownership/ITransferableOwnership.sol	b77b210598b25fdd7d6f309b7cc3bfc2f2aa6f68aec600144209795a47685114
infrastructure/ownership/Ownership.sol	8232872d78c68a10e5867c69f02a294a8ffd5e9871d6da44c86fd348f1e247cc

infrastructure/ownership/TransferableOwnership.sol	f0e062b0a74550d4e0d0e0a7f711ab6e4dd8dbb7e221e0a5bb70421a643c9a7c
integration/wings/IWingsAdapter.sol	7131e04a282aeb9fb251924f1701984e286da2a89f747b666c3af29e7aeb5c71
source/GLACrowdsale.sol	db8f33786561c163c26025dd41ba2110dee364753478dc6f653ca89450beeada
source/GLAToken.sol	5320db4ba397d3520559e1268638d43cfac29ccd3559888599717879191e0edc
source/crowdsale/Crowdsale.sol	9f86e0cd554fa4680957a6be7512022c04d2367894072836f243559b6090e0e1
source/crowdsale/ICrowdsale.sol	e97dfc7ed2f289716a73355fedc718971ab2feb2fdbcd3116a739d30584f18af
source/token/IManagedToken.sol	2d31a7c1268565a8d10f5b2cc09794f6306907d5662263963adc8d162342e816
source/token/IToken.sol	8da559d9c2e31d26677e885058e1d884faf302e79b5e60094735672cdc6a7d65
source/token/ManagedToken.sol	6684a728224d8daa42d384ef418ba99aa3ec1851b8156a853686f39004aeeac8
source/token/Token.sol	511eee3eb6993812011d4ff24698ebc18530e3e27d205035ec53855a753a83a7

A follow up audit was performed with the following files updated:

File	**Fingerprint (SHA256)**
source/GLACrowdsale.sol	25f7a040493cad85a46eb1b0788dfe2dcffe4a775e8ac38443a9b7362b456577
source/crowdsale/Crowdsale.sol	3a45da61c96bf7e8c4ac3ab69c1546ae67a9169e620ca5749d29801aeb9a535d
source/token/ManagedToken.sol	ce623a0ce308d99727391a831e22f1f10436c6ffbf2b375990d3f02d96306b13
source/token/Token.sol	13ccd606bb0fba0c65db2d3303717a90412d6a3d12b3273bc164b16ddb5f6c50

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

- Implements and adheres to existing ERC20 Token standard 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.

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of Gladius' token contract. To do so, 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.

Contract Analysis and Test Results

Summary

The Gladius token is a compliant ERC-20 Token with additional functionality added to adhere to the rules structured for their crowdsale.

During the audit several critical issues were discovered and promptly resolved by Frank Bonnet, the developer for Gladius's smart contract. We are glad that the Gladius team and Frank took into account our feedback on the issues presented. Overall, the Hosho team is pleased with the technical aspects of this contract, and other than the white paper discrepancies that are explained in greater detail in the Complete Analysis section, believe that this token and sale contract are extremely well written.

Coverage Report

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

Some lines are not under coverage due to the amount of mutual exclusion that comes from the way the contracts have been constructed. Many of the functions require time-shifting in order to achieve their full functionality and due to the limitations of the testing VM, it's not possible to combine all tests into one unified series. However, uncovered portions in this report, specifically, in regards to the `refund` function, which is the largest portion of the uncovered code, has been tested manually, as well in an alternate series of tests, and it is working as expected.

A number of the branches that are not covered are either impractical to hit as the system would need to be able to generate tokens in excess of the uint256 storage values, which is not possible with the published limits, or must match times exactly.

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

File	% Statements	% Branches	% Functions	% Lines
contracts/infrastructure/ITokenRetreiver.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/authentication/whitelist/IWhitelist.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/authentication/whitelist/whitelist.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/modifier/InputValidator.sol	100.00%	50.00%	100.00%	100.00%
contracts/infrastructure/modifier/Owned.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/ownership/IOwnership.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/ownership/ITransferableOwnership.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/ownership/Ownership.sol	100.00%	100.00%	100.00%	100.00%
contracts/infrastructure/ownership/TransferableOwnership.sol	100.00%	100.00%	100.00%	100.00%
contracts/integration/wings/IWingsAdapter.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/GLACrowdsale.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/GLAToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/crowdsale/Crowdsale.sol	92.40%	76.14%	96.77%	92.05%
contracts/source/crowdsale/ICrowdsale.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/token/IManagedToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/token/IToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/source/token/ManagedToken.sol	100.00%	75.00%	100.00%	100.00%
contracts/source/token/Token.sol	100.00%	80.00%	100.00%	100.00%
All files	94.65%	78.45%	98.48%	94.47%

Test Results

Contract: Gladius Crowdsale

- ✓ Amount of wei raised is correct;
- ✓ Should have a start that is before the end (87ms);
- ✓ Should have the presale timing set correctly;
- ✓ Should be not in presale phase before the `beneficiary` is set (48ms);
- ✓ Should not allow the crowdsale to be sent funds in a non `InProgress` stage (66ms);
- ✓ Should allocate tokens, and update amount raised (733ms);
- ✓ Should be in presale phase when starting to send funds (45ms);
- ✓ Should transfer external tokens from itself and the token contract to the owner (262ms);
- ✓ Should transfer external tokens from itself with the child being at 0 tokens (189ms);
- ✓ Should not transfer any tokens if both balances are 0. (114ms);
- ✓ Should not allow the crowdsale to be sent funds from non-whitelisted addresses (74ms);
- ✓ Should not allow the crowdsale to be sent funds under the minimum amount required during presale (192ms);
- ✓ Should allow the crowdsale to `init` and set balances (3474ms);
- ✓ Should not allow the crowdsale to be sent funds that would send it over the maximum presale amount during presale. (243ms);
- ✓ Should return correct balances for eth and tokens for the various stakeholders (137ms);
- ✓ Should not issue a refund until the crowdsale is over;
- ✓ Should be able to finalize the crowdsale and rate should become 0 (1497ms);
- ✓ Should not allow additional eth deposits once the crowdsale has ended. (50ms);
- ✓ Should allow presale eth to be withdrawn by the stakeholders immediately after the crowdsale has ended (540ms);
- ✓ Should not have any tokens available for immediate withdrawal by the stakeholders immediately after the crowdsale has ended (384ms);
- ✓ Should not let anyone but the `beneficiary` destroy the contract; and
- ✓ Should not let the `beneficiary` destroy the contract before 180 days have passed (38ms)

Contract: ERC-20 Compliant Token

- ✓ Should deploy with Gladius Token as the name of the token (55ms);
- ✓ Should deploy with GLA as the symbol of the token (82ms);
- ✓ Should deploy with 8 decimals;
- ✓ Should deploy with 0 tokens (40ms);
- ✓ Should allocate tokens per the minting function, and validate balances (5742ms);

- ✓ Should transfer tokens from 0xdd19f6e1365055565b7b2ec3586c2dcb5aa69d41 to 0xfc85a57470ac9a25e03b6400de95328245b8f94f (151ms);
- ✓ Should not transfer negative token amounts (75ms);
- ✓ Should not transfer more tokens than you have (62ms);
- ✓ Should allow 0x2a65494126a3642c150acbafa1ff1065a959d0d8 to authorize 0x18b16e186033bfde469ac8728ad95061e44c5b29 to transfer 1000 tokens (94ms);
- ✓ Should not allow 0x2a65494126a3642c150acbafa1ff1065a959d0d8 to authorize 0x18b16e186033bfde469ac8728ad95061e44c5b29 to transfer an additional 1000 tokens once authorized, and authorization balance is > 0 (75ms);
- ✓ Should allow 0x2a65494126a3642c150acbafa1ff1065a959d0d8 to zero out the 0x18b16e186033bfde469ac8728ad95061e44c5b29 authorization (87ms);
- ✓ Should allow 0x1fa129eae9a08f9ea5a7f43bab6038c1d27a573a to authorize 0x63247804c85ee1ffbc1cd10009c846872d5d79d1 for 1000 token spend, and 0x63247804c85ee1ffbc1cd10009c846872d5d79d1 should be able to send these tokens to 0x18b16e186033bfde469ac8728ad95061e44c5b29 (303ms);
- ✓ Should not allow 0x63247804c85ee1ffbc1cd10009c846872d5d79d1 to transfer negative tokens from 0x1fa129eae9a08f9ea5a7f43bab6038c1d27a573a (51ms); and
- ✓ Should not allow 0x63247804c85ee1ffbc1cd10009c846872d5d79d1 to transfer more tokens than permitted from 0x1fa129eae9a08f9ea5a7f43bab6038c1d27a573a (66ms)

Contract: Gladius Ownership

- ✓ Should return if someone is an owner or not; and
- ✓ Should return the contract owner

Contract: Gladius Token

- ✓ Should not allow eth to be sent to the contract (47ms);
- ✓ Should allow the crowdsale to `init` and set balances (4083ms);
- ✓ Should be locked against transfers to start;
- ✓ Should not permit transfers while locked; and
- ✓ Should `unlock` when the crowdSale ends (1498ms)

Contract: Gladius Whitelist

- ✓ Should allow someone to be added to the `whitelist`;
- ✓ Should not allow a non-owner to `whitelist` someone;
- ✓ Should allow someone to be removed from to the `whitelist`;
- ✓ Should allow someone to be removed from to the `whitelist` even if they aren't on it;
- ✓ Should not allow a non-owner to remove a `whitelist`; and
- ✓ Should allow someone to be re-added to the 14

Contract: Gladius Crowdsale Time-Shifting Tests

- ✓ Should allocate tokens, and update amount raised during presale (1490ms);
- ✓ Should exit presale once the first time-marker is hit (44ms);
- ✓ Should set whitelists in preparation to send funds (360ms);
- ✓ Should add refundable funds for a payment made after the first stage (413ms);
- ✓ Should not be able to finalize the crowdsale if the amount raised is too low;
- ✓ Should not allow excessively small payments inside of the ICO. (109ms);
- ✓ Should add refundable funds for a payment made after the first stage, and show as refundable (1437ms);
- ✓ Should be able to finalize the crowdsale and rate should become 0 (1423ms);
- ✓ Should return correct balances for eth and tokens for the various stakeholders (135ms);
- ✓ Should allow presale ETH to be withdrawn by the stakeholders immediately after the crowdsale has ended (574ms);
- ✓ Should not have any more eth to send after being drained (568ms);
- ✓ Should not have any tokens available for immediate withdraw by the stakeholders immediately after the crowdsale has ended (499ms);
- ✓ Should allow all token releases after 150 days (954ms); and
- ✓ Should let the beneficiary self-destruct the contract after 2 years (311ms)

Structure and Organization of Document

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.

Complete Analysis

Resolved, Critical: Presale Deposit Limitation

Explanation

Setting stakeholder ETH values above 0 causes the contract to error during token generation for deposit during the presale. This error doesn't appear under coverage which modifies gas limits heavily, but only shows up during normal test runs. As such, we suspect that this is related to gas.

There is a large amount of logic in the fallback function which should be shifted to a function that can access more than 2300 gas. This would most likely only be an issue during presale. The `_allocateStakeholdersEth` function accesses storage, which is not possible within the 2300 gas limit as defined by the contract at large. With values of ETH greater than 0, it calls the `_allocateEth` function, which then continues to use up the gas limit.

Solidity documentation on this subject can be read here:
http://solidity.readthedocs.io/en/develop/contracts.html#fallback-function

Resolution

The function in the fallback was split out to alleviate the concerns of the 2300 gas limit. This will work fine for using `send` normally, but breaks on using `send` from other contracts. Also a higher gas permitted path was added.

Resolved, Informational: Whitepaper Discrepancies

Explanation

The Hosho team did note some discrepancies between the WhitePaper, and the release of tokens. Specifically, the Whitepaper states the following:

> 10% of all GLA will go to the core founders.
>
> 10% will go the advisory, community, and marketing teams.
>
> 20% will go to operational costs, which includes bounty programs, day-to-day costs, etc.

Per the current deployment scripts, we show the following:

> 15% to dev team.
>
> 7.5% to "TLG"
>
> 7.5% to "Inbound"
>
> 10% to the bounty program.

Also, the whitepaper states that coins that don't go to the founding team will be available immediately. However, due to the design of the vesting code, the system releases 20% of the token value every 30 days, with 0 available at start.

Additionally, it's stated that the founding coins will be locked for 12 months, however, the longest lockout currently in the code is 150 days. Furthermore, we noted that the token sale coin distribution is targeted at 68M tokens, however, as there are 40% in scheduled bonuses, the total distribution should be 95.2M tokens, not the 96.3M in the WP, which is much closer to 42% bonuses.

Resolution

The Whitepaper has been updated to state:
10% of all GLA will go to the core founders.
15% will go to Advisory, Community, and Marketing teams
15% will go to operational costs

The deployment scripts have been updated to show the following:
10% to core founders

7.5% to TLG

7.5% to Inbound

13% to Bounty

2% to Wings.ai Community

The whitepaper has also been corrected to reflect the 150 day vesting periods and the 18 month founder lockout. We do still note that the token sale coin distribution is targeted at 68M tokens, however, as there are 40% in scheduled bonuses, the total distribution should be 95.2M tokens, not the 96.3M in the WP, which is much closer to 42% bonuses.

Resolved, Informational: Suggested Augmentation

source/crowdsale/Crowdsale.sol:L555

Explanation

The Hosho auditing team also suggests to put a block on the default `payable` function from contracts. Contracts cannot specify a gas amount when calling the internal `send` function.

Resolution

This was implemented by requiring `msg.sender` is the `tx.origin`.

Resolved, Informational: Ownership Note

Explanation

Ownership can never be changed on the token. It is locked to the Crowdsale forever once deployed.

Resolution

This is by design as it makes sure tokens are only issued during the crowdsale phase by the crowdsale.

Resolved, Informational: Unclear Intent

source/crowdsale/GLACrowdsale.sol:89

Explanation

In the crowdsale the function `retreiveTokens` is called by the `beneficiary`, but transfers to the `owner`. Since the tokens would be returned to the `owner`, it would be assumed that the `owner` would call this, not the `beneficiary`.

Resolution

Instead of `owner`, the function now returns to the `beneficiary` and continues to be called by the `beneficiary`.

Closing Statement

We are grateful to have been given the opportunity to work with the Gladius team and Frank Bonnet, the Solidity developer, whom we worked closely with in tackling the issues found during the audit. They were very receptive to our feedback and promptly resolved the issues brought forth.

As a small team of experts, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that Gladius' contracts are free of any glaring issues and that Frank has shown himself to be capable of removing issues quickly and effectively when presented the details.

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 Gladius Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.