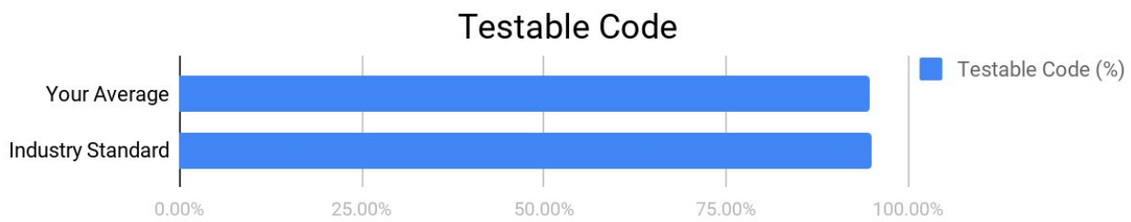
# BlitzPredict Token Contract Audit

by Hosho, March 2018

## Executive Summary

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


All issues have been remediated and suggestions added. (See Complete Analysis)




We are pleased to report that the testable code is on par with industry average. (See Coverage Report)

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 BlitzPredict 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 multiple reviews of the smart contract code, the latest version as written and updated on February 24, 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 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.
- Is not affected by the latest vulnerabilities

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of BlitzPredict's 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.

## 2. Structure Analysis and Test Results

### 2.1. Summary

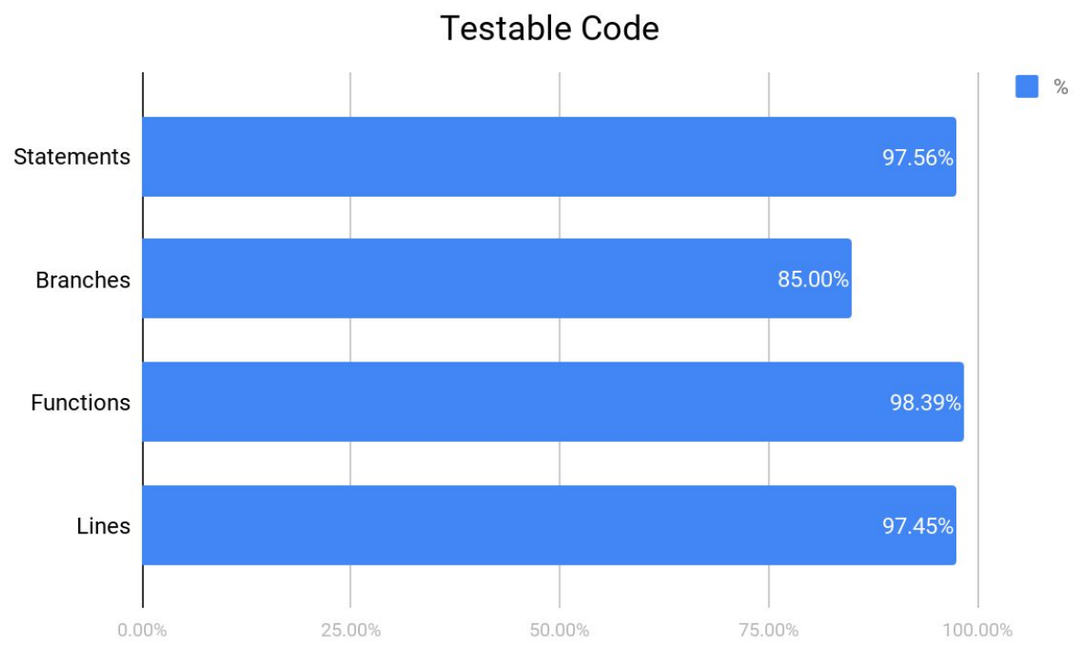
The BlitzPredict contract suite as reviewed by Hosho covers the following contracts: BPZSmartTokenSale, BPZSmartToken, TournamentManager, and VestingManager.

These contracts make up a standard Crowdsale, with no special tranche systems, making it very easy for possible investors to determine the token exchange rate as well as assists in ensuring that the system is simple and runs smoothly. The BPZ token is a simple ERC-20 token, with no particular limitations beyond an easily workable pausing system. Should transfers need to be paused, there is an administrative "destroy" function that can be used by the contract owner to destroy tokens belonging to any user. As part of the CrowdSale, the VestingManager contract controls and handles the long-term (1-4 year) storage of tokens for vesting purposes by various entities in the system.

The BlitzPredict contracts are fully testable line-by-line and we added tests to cover the new functionality such as whitelisting. These new functions caused coverage to slip slightly but all path were able to be manually tested and verified.

### 2.2 Coverage Report

As part of our work assisting BlitzPredict 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.

---

#### 7.1. Resolved, Critical: Token Issuance Error

BPZSmartTokenSale.sol

### Explanation

The function `getTokensPerEther` returns a value that is not the decimal value of the token leading to incorrect token values being issued. The result of this being that each participant receives a number of tokens that is vastly different than the purchased or transferred amount.

Technical Example: If the `etherPriceUSD` is set to 1, it should issue 5*10**19 tokens. Instead, given that msg.value is 1 ETH or 10**18 atomic units, the multiplication step becomes 10**18*5*10**19 then 5*10**37 and loops back to 3*10**26

### Resolution

The BlitzPredict Team resolved this issue by utilizing a fixed token issuance per ether rather than the `etherPriceUSD`.

---

#### 7.2. Resolved, Low: Continued Issuance After Crowdsale

BPZSmartToken.sol

### Explanation

The contract does not contain a mechanism to disable issuance once the crowdsale has been completed. This is suggested so that token holders can be assured there will be no tokens created after the close of the crowdsale.

### Resolution

Resolved by the BlitzPredict Team by adding two functions: `disableIssuance` and `disableDestruction`. While they are both tagged as enable and disable within the codebase they are strictly for disabling purposes and ensure that no tokens will be created or destroyed after the close of the crowdsale.

---

#### 7.3. Resolved, Low: Vesting Skip

VestingManager.sol

### Explanation

The contract contains a path by which the ability to skip the required vesting period can be acquired. The owner of the contract can call the `revokeGrant` function which revokes a grant and transfers the tokens to the owner. It is possible that, if the destinations are standard MultiSig wallets, the ownership could be transferred, allowing the grant to be revoked and immediately transferring the tokens, thereby circumventing the necessary vesting periods.

### Resolution

The `revokeGrant` functionality has been removed by the BlitzPredict Team, completely preventing the previously possible case of being able to circumvent the required vesting period.

---

#### 7.4. Resolved, Informational: Transfer On Destroy

SmartToken.sol

### Explanation

For the Transfer event on the `destroy` function, since the contract is not actually receiving these tokens, it may be better to use the void 0x0 address as the target for the event. While there is no specification regarding this, utilizing this pattern will be a simpler solution long term.

### Resolution

The Transfer event for the `destroy` function has been updated to utilize the 0x0 address by the BlitzPredict Team.

### 7.5. Resolved, Informational: ERC-20 Token Trap

#### Explanation

The Hosho team suggests adding an escape function for trapped tokens that are not issued by the contract. There are an increasing number of ERC-20 and ERC-223 tokens, such as the GOLEM token, getting trapped forever in contracts, so it is valuable to have a function that can return these tokens to a contract issuer or owner for refund.

#### Resolution

The file TokenRetriever.sol has been added by the BlitzPredict Team that adds token retrieval functionality to the contract, resolving the concern regarding trapped tokens.

## 4. Closing Statement

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

These BlitzPredict contracts follow a sound crowdsale pattern with additions to increase ease of use by investors as well as implementing a vesting schedule. These contracts are stable and offer a well written, testable codebase. The Hosho Team is pleased that the BlitzPredict Team has taken our suggested updates and implemented them quickly without causing any additional issues. The BlitzPredict Team has also been responsive and helpful in resolving any questions that arose and we look forward to working with them in the future.

All additional changes from February 24, 2018 are functioning as intended. As a small team of experts, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that the BlitzPredict 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 BlitzPredict Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

## 5. Test Suite Results

Contract: ERC-20 Compliant Token

✓ Should deploy with BlitzPredict as the name of the token (41ms)

✓ Should deploy with BPZ as the symbol of the token

✓ Should deploy with 18 decimals

✓ Should deploy with 0 tokens

✓ Should allocate tokens per the minting function, and validate balances (223ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (68ms)

✓ Should not transfer negative token amounts

✓ Should not transfer more tokens than you have

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (67ms)

✓ Should not allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer an additional 1000 tokens once authorized, and authorization balance is > 0 (45ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (48ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (138ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0

✓ Should not transfer tokens to 0x0

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

✓ Should allow an approval to be set, then increased, and decreased (173ms)

Contract: BlitzPredict Extended Tests

✓ Should allocate tokens per the minting function, and validate balances (223ms)

✓ Should not allow transfers to happen if transfers are disabled (48ms)

✓ Should allow the owner or the token holder to delete tokens, but shouldn't allow anyone else to do so (120ms)

✓ Should not allow tokens to be issued to the token contract (50ms)

✓ Should not allow tokens to be issued by anyone other than the contract owner (46ms)

✓ Should allow the transfer of ownership, and acceptance by the new owner only (229ms)

Contract: BlitzPredict Token Sale/Vesting Manager

✓ Should issue with the correct values for the number of tokens and round tokens, summing to 100% (180ms)

✓ Should not allow purchase before the crowdsale starts (92ms)

✓ Should allow the USD per ETH price to be set before the sale begins (223ms)

✓ Should not allow the USD per ETH price to be changed once the sale begins

✓ Should not allow funds to be sent directly to the contract (108ms)

✓ Should allow the owner of the token sale to transfer the ownership of the token contract (189ms)

✓ Should not allow the transfer of vestingManager ownership until the contract is finalized (86ms)

✓ Should not allow finalization if the contract has not had all tokens paid out via the crowdsale, or the end isn't hit

✓ Should not allow purchase with 0 wei (120ms)

✓ Should allow purchase, and issue the correct number of tokens for that purchase (599ms)

✓ Should allow finalization if the contract has had all tokens paid out via the crowdsale, even if the end isn't hit (290ms)

✓ Should allow the owner of the token sale to transfer the ownership of the vesting contract (172ms)

✓ Should not allow finalization if the contract is already finalized (56ms)

✓ Should not allow purchase after the crowdsale ends (88ms)

✓ Should allow the owner of the token sale to transfer the ownership of the vesting contract (68ms)

✓ Should not allow the owner to set a token grant that doesn't have enough tokens for it (42ms)

✓ Should not allow the owner to set a token grant on an already-existing grant

✓ Should have claimable tokens when the first time accelerations start (188ms)

✓ Should have tokens available after the first year, then incrementally more after that (1469ms)

✓ Should allow token reclaim (173ms)

Contract: BlitzPredict Tournament Manager

✓ Should not permit a contest to be published with an ID of 0

✓ Should permit a contest to be published with an ID of 1 (49ms)

✓ Should not permit a second contest to be published with an ID of 1

✓ Should not allow a request for a non-valid contest ID

✓ Should allow a request for a valid contest ID

✓ Should not allow a pick to be added if the signer isn't the owner (126ms)

✓ Should not allow a pick hash of 0 (89ms)

✓ Should allow a pick to be added if the signer is the owner (171ms)

✓ Should not allow someone to submit two picks (113ms)

✓ Should correctly handle no pick from an entrant

## 6. All Contract Files Tested

Original Files

| File | Fingerprint (SHA256) |
|---|---|
| contracts/BPZSmartToken.sol | fbdd7d3c1552cfb6d60658cde0bfa2851a75f740385d3dcbf5e52fac3835d4e0 |
| contracts/BPZSmartTokenSale.sol | 3791abfc070f9a1e6ed7b682f65a1c10571b33f0c1a5c2cde502af6beaadcd38 |
| contracts/TournamentManager.sol | 197088042375f97c92a6b1697cb650c1ab2bbfbba18c833d35df5187107002e8 |
| contracts/VestingManager.sol | 6346c780dcbade32d5ea0f668bc9889cd2f20f3bcf56967a03119c29b637adc4 |
| contracts/common/BaseContract.sol | 2e5778a887670b1f8433680354e9df6b30689de60141ebcffd473bab0c31752b |
| contracts/common/ERC20Token.sol | 4c6333a0397623b2e6d53503a5648b7b8a5a342f2d9f39958d34b803664b1d42 |
| contracts/common/Owned.sol | cf8be959e8a7d11d219346320c8e3bb0138f4cef809ab53716d32bd889babf43 |
| contracts/common/SafeMath.sol | 9b1744f4584d8a9c273055f20bbc462db543a74e5af34509a38f8308f1354885 |

Updated Files from January 9, 2018

| File | Fingerprint (SHA256) |
|---|---|
| contracts/BPZSmartToken.sol | 6057acc373fc73de6b602965032c684e27db5dc2a67c6910cc8b31b9d5acc119 |
| contracts/BPZSmartTokenSale.sol | 4fab01ac604933213e860ea5d511b671a68d01a9ea3f623efb4df525eb79111a |
| contracts/TournamentManager.sol | 197088042375f97c92a6b1697cb650c1ab2bbfbba18c833d35df5187107002e8 |
| contracts/VestingManager.sol | 9699b4036ac13a9fa911debd0e1c63fb4e4e0f53f8a62de849f11e00fd56db84 |
| contracts/common/BaseContract.sol | 2e5778a887670b1f8433680354e9df6b30689de60141ebcffd473bab0c31752b |
| contracts/common/ERC20Token.sol | 4c6333a0397623b2e6d53503a5648b7b8a5a342f2d9f39958d34b803664b1d42 |
| contracts/common/Owned.sol | cf8be959e8a7d11d219346320c8e3bb0138f4cef809ab53716d32bd889babf43 |
| contracts/common/TokenRetriever.sol | 6f79ccf97902d45660befd245104c101ee0992e6aaee2531c75c6ec425135f2a |

Updated Files from March 1, 2018

| File | Fingerprint (SHA256) |
|---|---|
| contracts/BPZSmartToken.sol | `296e26de711092f01ff6f095f5ad0bed5245bf0853c645f208936805a689ce32` |
| contracts/BPZSmartTokenSale.sol | `86e5eef499afd434d086712afa528ca69d8bd5c36ff4a9bda58ec5ac85507a8d` |
| contracts/TournamentManager.sol | `e13d430fa3c438f7e185c5a483e7afe20662814de33c8029361b2e99e4d65e0d` |
| contracts/VestingManager.sol | `bdcaccf50f6c7d90b5e581d58f3e6e0cd32bfcae0e3199d51422215e66a29100` |
| contracts/common/BaseContract.sol | `6bc21325e97875e796e0b3211b86fcf7054ea998493855d84ec53507220ca636` |
| contracts/common/ERC20Token.sol | `454a37d119dfdf9030f9bd7d77322b45a5f303b82748315d42a4b92baa740169` |
| contracts/common/IToken.sol | `7dc641d0712eef46945af3e5708afaa94c3013fe31b6a6bb3bfcd85a35a6846b` |
| contracts/common/Owned.sol | `cc38787ae097af29959848bb0f2e2a7e342c6c13a5e1e3df51dcf958caed3841` |
| contracts/common/SafeMath.sol | `f3d90a6edce5fadee6dd6928f5b7688c7e16198b206f130f9a16da4de1d79723` |
| contracts/common/SmartToken.sol | `d8b206cc79410974d979b5adf073cba772fb47fe1850556df120e3575c0f9c90` |
| contracts/common/TokenRetriever.sol | `c2b87d45d57f6324238a3acd900fdef5afd2b91d552a40c50f6ffe4199445943` |

## 7. Individual File Coverage Report

Original Files

| File | % Statements | % Branches | % Functions | % Lines |
|---|---|---|---|---|
| contracts/BPZSmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/BPZSmartTokenSale.sol | 100.00% | 72.22% | 100.00% | 100.00% |
| contracts/TournamentManager.sol | 100.00% | 85.71% | 100.00% | 100.00% |
| contracts/VestingManager.sol | 100.00% | 92.86% | 100.00% | 100.00% |
| contracts/common/BaseContract.sol | 100.00% | 88.89% | 100.00% | 100.00% |
| contracts/common/ERC20Token.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/Owned.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/SafeMath.sol | 100.00% | 66.67% | 100.00% | 100.00% |
| contracts/common/SmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| **All files** | **100.00%** | **85.00%** | **100.00%** | **100.00%** |

Updated Files from January 9, 2018

| File | % Statements | % Branches | % Functions | % Lines |
|---|---|---|---|---|
| contracts/BPZSmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/BPZSmartTokenSale.sol | 100.00% | 92.86% | 100.00% | 100.00% |
| contracts/TournamentManager.sol | 100.00% | 85.71% | 100.00% | 100.00% |
| contracts/VestingManager.sol | 90.01% | 68.75% | 85.71% | 91.18% |
| contracts/common/BaseContract.sol | 100.00% | 88.89% | 100.00% | 100.00% |

| | | | | |
|---|---|---|---|---|
| contracts/common/ERC20Token.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/IToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/Owned.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/SafeMath.sol | 100.00% | 66.67% | 100.00% | 100.00% |
| contracts/common/SmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/TokenRetriever.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| **All files** | **97.60%** | **82.50%** | **97.18%** | **97.99%** |

Updated Files from March 1, 2018

| **File** | **% Statements** | **% Branches** | **% Functions** | **% Lines** |
|---|---|---|---|---|
| contracts/BPZSmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/BPZSmartTokenSale.sol | 100.00% | 93.75% | 100.00% | 100.00% |
| contracts/TournamentManager.sol | 100.00% | 85.71% | 100.00% | 100.00% |
| contracts/VestingManager.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/BaseContract.sol | 100.00% | 85.71% | 100.00% | 100.00% |
| contracts/common/ERC20Token.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/IToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/Owned.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/SafeMath.sol | 100.00% | 66.67% | 100.00% | 100.00% |
| contracts/common/SmartToken.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| contracts/common/TokenRetriever.sol | 100.00% | 100.00% | 100.00% | 100.00% |
| **All files** | **97.56%** | **85.00%** | **98.39%** | **97.45%** |