PUBLIC

Expert Security Audit

of Smart Contracts

February 28, 2018

Produced by

CHAINSECURITY

for

PolicyPal Network

Table Of Content

Foreword

We first and foremost thank PolicyPal Network for giving us the opportunity to audit their smart contracts. This documents outlines our methodology, limitations, and results.

- ChainSecurity

Token Name & Symbol	POLICYPAL NETWORK TOKEN, PAL
Decimals	18 decimals
Phases	Presale, Crowdsale
Refund	None
Tokens issued	Set at contract creation
Minimum contribution	Fixed at creation
Maximum contribution	Set at creation & with fixed change
Token Type	Utility Token
Token Generation	Pre-minted
Vesting	None
Pausable	Available
Whitelist	Add & Remove
Owner Rewards	50% of PALs

Table 1: Facts about the PAL token and the token sale.

System Overview

POLICYPAL NETWORK aims to offer access to insurance protection in a wide range of industries through an Ethereum-based platform and a network of related applications within its own ecosystem. Holders of POLICYPAL NETWORK TOKENS will be able to use it for services provided by the POLICYPAL NETWORK, like purchasing insurance policies or survey participation to express interest for new product features or development directions.

In the following we describe the POLICYPAL NETWORK TOKEN (PAL) and its corresponding token sale. Table 1 gives the general overview.

Token Sale Overview

After the tokens are minted by the contract owner, he can transfer the contract ownership to an administrator. The crowdsale proceeds within a predefined time range from a start- to an end-date during which tokens can be purchased for ETH. The token sale can be halted by the administrator at any time.

Token Overview

The token audited is a ERC20 token, fulfilling the functionalities defined by this standard. Its supply is fixed, it can be burnt, and the distribution of tokens in between the reserve, presale and the ongoing crowdsale will be transparent and can be observed on an Etherscan holders card. The token will be purchasable by whitelisted customers who can buy up to a maximum contribution threshold. After a pre-defined time, called `increaseMaxContribTime` the maximum contribution threshold will be raised by the factor of 10.

Extra Token Features

Haltable The token contract can be paused and unpaused by the contract administrator, arbitrarily often. While the token is Halted only the administrator and the crowdsale contract can make token transfers.

Emergency Drain The Administrator of the contract can drain a variable amount of funds in case something went wrong and ETH is stuck in the contract.

Burnable Tokens can be burnt, a feature which, according to the white paper, will be used from the third year onwards. This feature is available to all token owners.

Audit Overview

Scope of the Audit

The scope of the audit is limited to the following source code files. All of these source code file were received on February 28, 2018:

File	SHA-256 checksum
CrowdsaleAuthorizer.sol	9e9bb3fc46f16d14b89d5b97d534e7fce289f65df6a1fe507067016ad75f2b61
PolicyPalNetworkCrowdsale.sol	8aed46069760fd8a40571fde3280e22a7c21411a9a5e21e42a76b6d4bd3923bf
PolicyPalNetworkToken.sol	2d9c22b87b115d6cde4486f060cddc24c617f8f4fbf8e03b95815099f276e239
zeppelin/math/SafeMath.sol	e434336813af116101008bcfaed8cc02fa051c9c2b612a477b6bfa0765fa17f6
zeppelin/ownership/Ownable.sol	35dcf237365077adb1dd8d1da9e05f2b4f8e9d7b49311fc8a09b28d4ce191579
zeppelin/token/BasicToken.sol	7bc1695b8a0ab00d5795d930f9597e0553d8f6ab2e47d87386fbe6488d472bb5
zeppelin/token/BurnableToken.sol	6ae07ae6cc71af27e29af199dbfc2d97f28fcc427398d88cd57eddd456e7a351
zeppelin/token/ERC20.sol	6b75acd05c29968b057ec1facf659c064dbe0a79ac01444530629f01ef3a3abf
zeppelin/token/ERC20Basic.sol	86c0a5fc6cb564ae77140da57a8ff9a22f46404240e69a6782ff741e286d373a
zeppelin/token/StandardToken.sol	77e45da1164753f886d7395987b46deb036eca32c2e7322ef7a2764a08f7c5da

Depth of Audit

The scope of the security audit conducted by CHAINSECURITY was restricted to:

- Scan the contracts listed above for generic security issues using automated systems and manually inspect the results.
- Manual audit of the contracts listed above for security issues.

Terminology

For the purpose of this audit, we adopt the following terminology. For security vulnerabilities, we specify the *likelihood*, *impact* and *severity* (inspired by the OWASP risk rating methodology[1]).

Likelihood represents the likelihood of a security vulnerability to be encountered or exploited in the wild.

Impact specifies the technical and business related consequences of an exploit.

Severity is derived based on the likelihood and the impact calculated previously.

We categorize the findings into 4 distinct categories, depending on their criticality:

- L - can be considered as less important
- M - needs to be considered to be fixed
- H - is strongly suggested to be fixed
- C - needs to be fixed before the deployment

[1] https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

	IMPACT		
LIKELIHOOD	High	Medium	Low
High	C	H	M
Medium	H	M	L
Low	M	L	L

During the audit concerns might arise or tools might flag certain security issues. If our careful inspection reveals no security impact, we label it as ✓ No Issue. Finally, if during the course of the audit process, an issue has been addressed technically, we label it as ✓ Fixed, while if it has been addressed otherwise we label it as ✓ Addressed.

Findings that are labelled as either ✓ Fixed or ✓ Addressed are resolved and therefore pose no security threat. Their severity is still listed, but just to give the reader a quick overview what kind of issues were found during the audit.

Limitations

Security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a secure smart contract. However, auditing allows to discover vulnerabilities that were overlooked during development and areas where additional security measures are necessary.

In most cases, applications are either fully protected against a certain type of attack, or they lack protection against it completely. Some of the issues may affect the entire smart contract application, while some lack protection only in certain areas. We therefore carry out a source code review trying to determine all locations that need to be fixed. Within the customer-determined timeframe, CHAINSECURITY has performed auditing in order to discover as many vulnerabilities as possible.

Details of the Findings

In this section we detail our findings, including both positive and negative findings.

Confusion between token balance and ether balance M ✓ Fixed

The crowdsale contract keeps track of the amount each participant can still contribute as `weiContributionAllowed` and the remaining tokens as `tokensRemaining`. However, these two should not directly be compared, as they have to use the conversion rate. In the following code, this mistake is made:

PolicyPalNetworkCrowdsale.sol

```
        // Get tokens remaining for sale
        uint256 tokensRemaining = token.balanceOf(address(this));
        require(tokensRemaining > 0);

        if (weiContributionAllowed > tokensRemaining) {
            weiContributionAllowed = tokensRemaining;
```

As a consequence an otherwise valid token purchase can fail and it becomes impossible to sell out completely.

Likelihood: Low
Impact: High

Post-Audit Fix: PolicyPal Network has fixed this issue by converting the remaining tokens into their wei value and then comparing different wei values as can be seen in the code below:

PolicyPalNetworkCrowdsale.sol

```
        // Convert tokens to wei
        uint256 tokensRemainingInWei = tokensRemaining.div(rate);

        if (weiContributionAllowed > tokensRemainingInWei) {
            weiContributionAllowed = tokensRemainingInWei;
        }
```

Constant-Gas Reentrancy M ✓ Fixed

There are two possible constant-gas reentrancies within the `buy` function of the crowdsale. The more critical one is:

PolicyPalNetworkCrowdsale.sol

```
        // Check weiContributionAllowed is larger than value sent
        // If larger, transfer the excess back to the contributor
        if (msg.value > weiContributionAllowed) {
            msg.sender.transfer(msg.value.sub(weiContributionAllowed));
        }
```

Here, untrusted code is invoked while the contract is in an inconsistent state. Therefore, this operation should be performed at the end of the `buy` function.

Additionally, there is another occurrence here:

PolicyPalNetworkCrowdsale.sol

```
        // Send ETH payment to MultiSig Wallet
        sendETHToMultiSig(weiContributionAllowed);
        raisedWei = raisedWei.add(weiContributionAllowed);
```

Here, the transfer is made to a trusted account, which is not as critical, but should still happen in an otherwise consistent state at the end of the function.

Likelihood: Low
Impact: High

Post-Audit Fix: PolicyPal Network has fixed this issue by moving the refunding ether transfer to the end of the `buy` function, when the contract is in a consistent state. This prevents reentrancy attacks.

Potentially Empty Token Purchases L ✓ Fixed

PolicyPal Network is dividing the number of remaining tokens by the conversion rate to find the value of the remaining tokens in wei:

PolicyPalNetworkCrowdsale.sol

```
        // Convert tokens to wei
        uint256 tokensRemainingInWei = tokensRemaining.div(rate);
```

However, the result of this division might be 0 if only a small number of tokens are left. In this case the `buy` function simply continues to execute and performs an empty token purchase.
Likelihood: Medium
Impact: Low

Post-Audit Fix: PolicyPal Network has fixed this issue by refactoring the associated crowdsale code. In particular, the critical line mentioned above is not included any longer.

Potentially no sellout L ✓ Fixed

Depending on the set `rate` and the contributed amount, it might not be possible to sell out all of the tokens. If one token buyer tries to buy all of the remaining tokens, but the number cannot be evenly divided by the `rate`, then the residual tokens will not be given to the token buyer and can also not be bought by any other token buyer (see issue above). The impact of this is low as this would concern at most $(\texttt{rate} - 1) \cdot 10^{-18}$ PALs.
Likelihood: Medium
Impact: Low

Post-Audit Fix: PolicyPal Network has fixed this issue by assigning all remaining token fractions to the token buyer that is trying to reach a sellout. The associated code is:

PolicyPalNetworkCrowdsale.sol

```
            // Check remaining tokens
            // If lesser, update tokens to be transfer and contribution allowed
            if (receivedTokens > tokensRemaining) {
                receivedTokens = tokensRemaining;
                weiContributionAllowed = tokensRemaining.div(rate);
            }
```

No Callstack Bugs ✓ No Issue

We did not discover any callstack issues.

Ether Transfers ✓ No Issue

We did not discover unusual or dangerous ether transfers in the code as the PAL contract directly forwards received ether to a specified wallet.

Safe Math ✓ No Issue

The PolicyPal Network contracts use the safe math library to avoid over-/under-flows. In particular, critical variables such as `raisedWei` and the `balances` are always handled with calls to the safe math library.

Unused Variables ✓ Fixed

The `saleStartTime` and `saleEndTime` variables inside `PolicyPalNetworkToken` are unused and could be removed.

Recommendations / Suggestions ✓ Fixed

- The `admin` has a very powerful role as it has access to the restricted functionalities. Therefore, you might consider a recovery option to transfer the `admin` role.
- It is not clear whether the `setMaxContribution` function is needed. Its functional use is limited and due to its nature, it leads to race conditions and might therefore cause unexpected transaction failures.
- The `Claimable` library can replace the `Ownable` library to prevent issues during ownership transfer.
- Optionally, you can use some constants in the constructor instead of making it all configurable in case the numbers are anyways fixed. This reduces the deployment complexity and improves readability.
- You could add the following check to CrowdsaleAuthorizer to avoid mistakes:

Optional Improvement

```
        require(_saleStartTime > now);
```

- Some functions such as `updateWhitelists` can be labelled as `external` in order to save some gas.
- Some test cases are asynchronous so cannot report errors. Add async and await everywhere.
- For the efficient and fast distribution of tokens that were purchased in the presale, it might be beneficial to add a batch-transfer function, which performs multiple transfers together. This could reduce transaction costs for POLICYPAL NETWORK.
- Seemingly incorrect/confusing comments:

PolicyPalNetworkToken.sol

```
        // Transfer half the supply from token creator to admin
        transferOwnership(_adminAddr);
```

CrowdsaleAuthorizer.sol

```
     * @param _premintedTokenSupply - Total preminted token supply
```

PolicyPalNetworkCrowdsale.sol

```
        // Return either the amount contributed or cap whichever is lower
```

- The `updateWhitelists` function calls the `updateWhitelist` function repeatedly. As a consequence the modifiers of `updateWhitelist` are checked multiple times. This leaves room for optimizations as gas costs can be lowered.
- The supply with 18 decimals exceeds the precision of JavaScript integer representation. `BigNumber.js` should be used in future scripts and applications.
- The following check should also take the value of `_premintedTokenSupply` into account.

PolicyPalNetworkCrowdsale.sol

```
require(_premintedTokenSupply < _totalTokenSupply);
```

- The `multiSigWallet` has to be able to perform token transfers. Otherwise, these tokens will be lost.
- Block timestamps can be manipulated to a limited degree by the miners. However, this is not an issue in the way they are used in these contracts.

Post-audit comment: POLICYPAL NETWORK has implemented a major part of the recommendations. This provide more control, assurance and also provide minor gas savings.

Conclusion

ChainSecurity has intensively audited the PolicyPal Network Token smart contracts using research-driven, in-house analysis tools as well as manual analysis by experts. The token and the token sale have the required functionality and after the good cooperation of PolicyPal Network and ChainSecurity, all detected issues were fixed. Hence, there are no remaining security issues from ChainSecurity's perspective.

Disclaimer