BUG REPORT : tGNT Faucet Can Easily Be Emptied

Author : Philippe Castonguay

Date : 11.04.2018

Repository : https://github.com/PhABC/golem-contracts/tree/faucetVacuum

Affects: All contracts on deployed on Rinkeby (or other test contracts)

Short Description : The testGNT faucet can easily be emptied (cost ~76 Rinkeby ETH)

Description

The **Faucet.sol** contract design makes it relatively easy for an attacker to empty the **tGNT** faucet on Rinkeby (https://rinkeby.etherscan.io/token/0x924442a66cfd812308791872c4b242440c108e19?a=0x77b6145e853dfa80e8755a4e824c4f510ac6692e). Indeed, an attacker can simply write a wrapper contract that calls the `create()` function on the **Faucet** contract, transfer the tGNT to a recipient address (e.g. `0x0`) and then repeat the process iteratively. The fault is also due to the **small** and **fixed** token supply of tGNT, which is 1 billion (https://rinkeby.etherscan.io/token/0x924442a66cfd812308791872c4b242440c108e19).

This problem may seem trivial, but could lead to some complications, since all Golem contracts **can not update which GNT token contract they are using**. Hence, if the faucet was emptied, all contracts on Rinkeby would need to be redeployed and users will need to resend various transactions. All deposits, opened channels and other contract states would need to be remade. This would be time consuming, both for the Golem team and testers.

This will flaw will become even more important in the future, where a large network of Golem service providers and users will actively test new Golem versions before they are released on the mainnet, having a financial incentive to do so. This is not even considering if Golem goes mainstream as a computing infrastructure. Hence, the earlier this attack vector is addressed, the smaller the consequences on the Golem testing network and infrastructures.

Attack scenario

Bob is a competitor and wants to impair the Golem team and all their testers by forcing a redeployment of the contracts on the testnet. Bob creates an attacker contract that empties the Faucet, forcing all contracts to be redeployed.

Impact: Low

All test contracts (e.g. on Rinkeby) would need to be redeployed to allow new supply of tokens to fund the faucet.

Difficulty : Easy

Cost is between `38` and `114` **Rinkeby ETH** (see below).

Components:

- **All contracts deployed on Rinkeby** (and other test contracts)

Reproduction:

Github Repository : https://github.com/PhABC/golem-contracts/tree/faucetVacuum

Instructions :

```
npm install
npm test
```

FaucetVacuum Contract : https://github.com/PhABC/golem-contracts/blob/faucetVacuum/contracts/FaucetVacuum.sol

FaucetVacuum Test Script : https://github.com/PhABC/golem-contracts/blob/faucetVacuum/test/faucetVacuuming.test.js

Details:

The attacker's contract contains the following function ;

```
function vacuum(uint32 _nIterations) public {
    for(uint32 i = 0; i < _nIterations; i++){
        faucet.create();                      // Claims tokens from faucet
        token.transfer(0x0, creationAmount);  // Destroy tokens
    }
}
```

The Rinkeby gas limit is `6999661` which easily allows for more than `100` iterations per transaction. Indeed, with `100` iterations, the `gasUsed` is about `3804495`. Hence, the total cost to take `1,000,000,000` **tGNT** from a faucet would be between `38` and `114` Rinkeby **ETH**, assuming a `gasPrice` between 1-3 **Gwei**. Cost could be lowered with lower level optimization.

Fixes

Simple fixes are numerous :

1. Set a ETH cost to call the faucet (e.g. 0.1 ETH) per `create()` call.
2. Reduce number of tGNT sent per `create()` call (e.g. 1, value is arbitrary when testing).
3. Used an authenticated faucet (e.g. https://faucet.rinkeby.io/).
4. Increase the **tGNT** total supply dramatically, like `2**256`. (see the Rinkeby ETH faucet as an example ; https://rinkeby.etherscan.io/address/0x31b98d14007bdee637298086988a0bbd31184523). **Requires redeploy of all contracts**
5. ...

However, all these proposals, **at the very least**, require to transfer the **tGNT** tokens to a **new** faucet contract. Since the current **Faucet.sol** implementation can't easily transfer tokens out, the attacker contract provided in this report could be used as a mean to transfer the tGNT to a new faucet contract. This would prevent the need from redeploying any contract and would avoid any testing disruption. Redeployment of all contracts is also an option.