Gelato Ops Audit Report

Aug 16, 2022

Table of Contents

Summary

This report has been prepared for Gelato Ops Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**Gelato Ops**
Codebase	**https://github.com/gelatodigital/ops**
Commit	**8722d188b367f8bdeac6b6daadae7a834f12cb5b**
Language	**Solidity**

Audit Summary

Delivery Date	**Aug 16, 2022**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**5**

[WP-L1] Obsoleted variables and methods after the onlyOneProxy change

Low

Issue Description

1. `_proxies[proxy]` can be replaced with `_ownerOf[proxy]` :

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L87-L103

```
 88  function deployFor(address owner)
 89      public
 90      override
 91      onlyOneProxy(owner)
 92      notProxy(owner)
 93      returns (address payable proxy)
 94  {
 95      (bytes32 seed, bytes32 salt) = _getSeedAndSalt(owner);
 96
 97      bytes memory bytecode = _getBytecode(owner);
 98
 99      proxy = _deploy(salt, bytecode);
100
101      _proxies[proxy] = true;
102      _proxyOf[owner] = proxy;
103      _ownerOf[proxy] = owner;
```

L101 can be removed.

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L139-L141

```
139      function isProxy(address proxy) public view override returns (bool) {
140          return _proxies[proxy];
141      }
```

`isProxy()` can be changed to:

```
    function isProxy(address proxy) public view override returns (bool) {
        return _ownerOf[proxy] != address(0);
    }
```

`opsProxyFactory.isProxy(account)` can be replaced with `opsProxyFactory._ownerOf(account)` :

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/taskModules/ProxyModule.sol#L37-L69

```
function preCreateTask(address _taskCreator, address _execAddress)
    external
    view
    override
    returns (address, address)
{
    bool isExecAddressProxy = opsProxyFactory.isProxy(_execAddress);

    if (isExecAddressProxy) {
        address ownerOfExecAddress = opsProxyFactory.getOwnerOf(
            _execAddress
        );
        require(
            _taskCreator == ownerOfExecAddress ||
                _taskCreator == _execAddress,
            "ProxyModule: Only owner of proxy"
        );

        return (ownerOfExecAddress, _execAddress);
    } else {
        bool isTaskCreatorProxy = opsProxyFactory.isProxy(_taskCreator);

        if (isTaskCreatorProxy) {
            address ownerOfTaskCreator = opsProxyFactory.getOwnerOf(
                _taskCreator
            );

            return (ownerOfTaskCreator, _execAddress);
        }

```

```
            return (_taskCreator, _execAddress);
        }
    }
```

Recommendation

1. Remove `getOwnerOf()` and rename `_ownerOf` to `ownerOf` and make it `public` ;
2. Change `preCreateTask()` to:

```
function preCreateTask(address _taskCreator, address _execAddress)
    external
    view
    override
    returns (address, address)
{
    address ownerOfExecAddress = opsProxyFactory.ownerOf(
        _execAddress
    );

    if (ownerOfExecAddress != address(0)) {
        require(
            _taskCreator == ownerOfExecAddress ||
                _taskCreator == _execAddress,
            "ProxyModule: Only owner of proxy"
        );

        return (ownerOfExecAddress, _execAddress);
    } else {
        address ownerOfTaskCreator = opsProxyFactory.ownerOf(
            _taskCreator
        );

        if (ownerOfTaskCreator != address(0)) {
            return (ownerOfTaskCreator, _execAddress);
        }

        return (_taskCreator, _execAddress);
    }
}
```

2. `_nextSeeds` is no longer needed:

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L155-L163

```
function _getSeedAndSalt(address _account)
    internal
    view
    returns (bytes32 seed, bytes32 salt)
{
    seed = _nextSeeds[_account];

    salt = keccak256(abi.encode(_account, seed));
}
```

`seed` can only be `0` and `salt` will always be `keccak256(abi.encode(_account, 0))` as only one proxy is allowed for one `account` .

Therefore, `deployFor()` can be changed to:

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L88-L110

```
function deployFor(address owner)
    public
    override
    onlyOneProxy(owner)
    notProxy(owner)
    returns (address payable proxy)
{
    proxy = _deploy(bytes32(uint256(uint160(owner))), _getBytecode(owner));

    _proxyOf[owner] = proxy;
    _ownerOf[proxy] = owner;

    emit DeployProxy(msg.sender, owner, address(proxy));
}
```

`determineProxyAddress()` should also be changed accordingly.

`getNextSeed()` , `_getSeedAndSalt()` can be removed.

Using `bytes32(owner)` may also be unnecessary, could just use `bytes32(0)` as the salt.

Status

✓ Fixed

[WP-L2] Implementation should be whitelisted to safeguard users from hijacking attack

Low

Issue Description

The current implementation allows the owner of the `OpsProxy` to call `upgradeTo()` and upgrade to an arbitrary new implementation:

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/vendor/proxy/EIP173/EIP173Proxy.sol#L66-L68

```
function upgradeTo(address newImplementation) external onlyProxyAdmin {
    _setImplementation(newImplementation, "");
}
```

This could be a problem if the attacker managed to hijack the frontend and deceived the user into calling `upgradeTo()` and set to a malicious implementation.

Recommendation

Consider overriding the `upgradeTo()` and `upgradeToAndCall()` , only allows `_setImplementation` to a whitelisted implementation on `OpsProxyFactory.sol` .

Status

✓ Fixed

[WP-L3] Empty implementation of TaskModuleBase#preCreateTask() is error-prone

Low

Issue Description

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/taskModules/TaskModuleBase.sol#L12-L17

```
    function preCreateTask(address, address)
        external
        virtual
        override
        returns (address, address)
    {}
```

The default empty implementation will return `address(0), address(0)` as the `taskCreator` and `execAddress` .

If configurated correctly, the empty implementation of `preCreateTask()` will not be called unless `module.requirePreCreate() == true` :

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/libraries/LibTaskModule.sol#L23-L56

```
    function preCreateTask(
        address _taskCreator,
        address _execAddress,
        mapping(LibDataTypes.Module => address) storage taskModuleAddresses
    ) internal returns (address, address) {
        uint256 length = uint256(type(LibDataTypes.Module).max);

        for (uint256 i; i <= length; i++) {
            LibDataTypes.Module module = LibDataTypes.Module(i);
            if (!module.requirePreCreate()) continue;

            address moduleAddress = taskModuleAddresses[module];
            _moduleInitialised(moduleAddress);

            bytes memory delegatecallData = abi.encodeWithSelector(
                ITaskModule.preCreateTask.selector,
                _taskCreator,
                _execAddress
            );

            (, bytes memory returnData) = _delegateCall(
                moduleAddress,
                delegatecallData,
                "Ops.preCreateTask: "
            );

            (_taskCreator, _execAddress) = abi.decode(
                returnData,
                (address, address)
            );
        }

        return (_taskCreator, _execAddress);
    }
```

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/Ops.sol#L46-L67

```
    function createTask(
        address _execAddress,
        bytes calldata _execDataOrSelector,
        LibDataTypes.ModuleData calldata _moduleData,
        address _feeToken
    ) external override returns (bytes32 taskId) {
        address taskCreator;

        (taskCreator, _execAddress) = LibTaskModule.preCreateTask(
            msg.sender,
            _execAddress,
            taskModuleAddresses
        );

        taskId = _createTask(
            taskCreator,
            _execAddress,
            _execDataOrSelector,
            _moduleData,
            _feeToken
        );
    }
```

We believe it will be less error-prone if it reverts instead of returning `address(0), address(0)` .

Recommendation

Consider changing to:

```
function preCreateTask(address, address)
    external
    virtual
    override
    returns (address, address)
{
    revert("Not Implemented");
}
```

Status

✓ **Fixed**

[WP-I4] OpsProxyFactory.notProxy(owner) can be bypassed

Informational

Issue Description

While there is a `notProxy(owner)` modifier on `deployFor()` , this restriction can be bypassed by creating the proxy of a proxy before creating that proxy:

1. Query the `determineProxyAddress()` for the `account` ;
2. `deployFor()` with the address above as the `owner` ;
3. `deployFor()` with `account` as the `owner` .

This may or may not be a problem depending on what `notProxy(owner)` was designed for.

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L88-L110

```
function deployFor(address owner)
    public
    override
    onlyOneProxy(owner)
    notProxy(owner)
    returns (address payable proxy)
{
    (bytes32 seed, bytes32 salt) = _getSeedAndSalt(owner);

    bytes memory bytecode = _getBytecode(owner);

    proxy = _deploy(salt, bytecode);

    _proxies[proxy] = true;
    _proxyOf[owner] = proxy;
    _ownerOf[proxy] = owner;

    unchecked {
        _nextSeeds[owner] = bytes32(uint256(seed) + 1);
    }

    emit DeployProxy(msg.sender, owner, seed, salt, address(proxy));
}
```

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L35-L38

```
modifier notProxy(address _account) {
    require(!isProxy(_account), "OpsProxyFactory: No proxy");
    _;
}
```

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxyFactory.sol#L139-L141

```
function isProxy(address proxy) public view override returns (bool) {
    return _proxies[proxy];
}
```

Status

ⓘ Acknowledged

[WP-I5] The restriction of only the owner of the OpsProxy can create a task that calls the OpsProxy can be bypassed

Informational

Issue Description

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/taskModules/ProxyModule.sol#L37-L69

```
function preCreateTask(address _taskCreator, address _execAddress)
    external
    view
    override
    returns (address, address)
{
    bool isExecAddressProxy = opsProxyFactory.isProxy(_execAddress);

    if (isExecAddressProxy) {
        address ownerOfExecAddress = opsProxyFactory.getOwnerOf(
            _execAddress
        );
        require(
            _taskCreator == ownerOfExecAddress ||
                _taskCreator == _execAddress,
            "ProxyModule: Only owner of proxy"
        );

        return (ownerOfExecAddress, _execAddress);
    } else {
        bool isTaskCreatorProxy = opsProxyFactory.isProxy(_taskCreator);

        if (isTaskCreatorProxy) {
            address ownerOfTaskCreator = opsProxyFactory.getOwnerOf(
                _taskCreator
            );

            return (ownerOfTaskCreator, _execAddress);
        }

        return (_taskCreator, _execAddress);
    }
}
```

By front-running a proxy creating transaction (`deployFor()`) and create a task for that address, which is not yet deployed, this restriction can be bypassed.

Because the `OpsProxyFactory._proxies[proxy]` is not yet set to `true` , it will not go into the branch of L45-56.

Thanks to the check on OpsProxy, we believe there is no way to exploit it though:

https://github.com/gelatodigital/ops/blob/5524495a6864c51fc6479b04800278977a2e0373/contracts/opsProxy/OpsProxy.sol#L16-L31

```
modifier onlyAuth() {
    require(
        msg.sender == ops || msg.sender == owner(),
        "OpsProxy: Not authorised"
    );

    if (msg.sender == ops) {
        address taskCreator = _getTaskCreator();

        require(
            taskCreator == owner(),
            "OpsProxy: Only tasks created by owner"
        );
    }
    _;
}
```

Status

ⓘ Acknowledged

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.