CoW-Shed Audit

This document presents the finding of a smart contract audit conducted by Côme du Crest for Gnosis.

Scope

The initial scope includes all contracts within `src/` of cow-shed as of commit 853206c. The audit fixes have been implemented in commit https://github.com/cowdao-grants/cow-shed/commit/63eeff3c586ba6220a2707c20542cac7d019e97a and an opt-out feature for ENS has been implemented in https://github.com/cowdao-grants/cow-shed/commit/0f7eaf419797d1af96cb3a9d027f2596f15521ae. Both of these commits have been reviewed as well.

Context

The idea behind cow-shed is to deploy an ERC1967 proxy at a deterministic address for a user to execute post or pre-hooks on CoW swap orders. The proxy can execute arbitrary calls authorized by the user via a signature before or after a CoW swap.

Status

The report has been reviewed by developers and fixes implemented. No issue remain.

Issues

▼ [Low] Owner/Operator of baseNode can deny proxy deployment

Summary

The owner/operator of `baseNode` on ENS can deny proxy deployment by removing the authorization of `baseNode` from the `COWShedFactory`.

Vulnerability Detail

During deployment on mainnet, the proxy factory will register the proxy on ENS using `ENS.setSubnodeRecord()`:

```
contract COWShedFactory is COWShedResolver {
    ...

    /// @notice execute hooks on user proxy
    /// @dev Will deploy and initialize the user proxy at a deterministic address
    ///      if one doesn't already exist.
    function executeHooks(
        Call[] calldata calls,
        bytes32 nonce,
        uint256 deadline,
        address user,
        bytes calldata signature
    ) external {
        address proxy = proxyOf(user);
        // deploy and initialize proxy if it doesnt exist
        if (proxy.code.length == 0) {
            ...
            // if on mainnet, set the forward and reverse resolution nodes
```

```
            if (block.chainid == 1) {
                _setReverseNode(user, proxy);
                _setForwardNode(user, proxy);
            }
        }
        ...
    }
}

abstract contract COWShedResolver is INameResolver, IAddrResolver {

        ...
    function _setForwardNode(address user, address proxy) internal {
        _setForwardNodeForAddressString(LibString.toHexStringChecksummed(user), p
        _setForwardNodeForAddressString(LibString.toHexString(user), proxy);
    }

    function _setForwardNodeForAddressString(string memory labelString, address p
        bytes32 label = keccak256(abi.encodePacked(bytes(labelString)));
        ENS.setSubnodeRecord(baseNode, label, address(this), address(this), type(
        ...
    }
}
```

The call to `ENS.setSubnodeRecord(baseNode ...)` requires that `msg.sender` be authorized on `baseNode`:

```
contract ENSRegistry is ENS {
    ...
    modifier authorised(bytes32 node) {
        address owner = records[node].owner;
        require(owner == msg.sender || operators[owner][msg.sender]);
        _;
    }

    function setSubnodeRecord(
        bytes32 node,
        bytes32 label,
        address owner,
        address resolver,
        uint64 ttl
    ) external virtual override {
        bytes32 subnode = setSubnodeOwner(node, label, owner);
        _setResolverAndTTL(subnode, resolver, ttl);
    }

    function setSubnodeOwner(
        bytes32 node,
        bytes32 label,
        address owner
    ) public virtual override authorised(node) returns (bytes32) {  // @audit req
        bytes32 subnode = keccak256(abi.encodePacked(node, label));
        _setOwner(subnode, owner);
```

```
        emit NewOwner(node, label, owner);
        return subnode;
    }
}
```

This means that the operator or owner of `baseNode` can remove the rights to call `setSubnodeRecord()` for the `COWShedFactory` , in which case the call will revert and proxy deployment will fail.

Impact

On mainnet, the operator or owner of `baseNode` can selectively deny deployment of proxies. If the proxy address was pre-computed and funds were sent to it, the funds may be frozen indefinitely.

Code Snippets

https://github.com/cowdao-grants/cow-shed/blob/853206cd8ec43efbddda1bec7f2adfd19cd7920d/src/COWShedFactory.sol#L47-L48

https://github.com/cowdao-grants/cow-shed/blob/853206cd8ec43efbddda1bec7f2adfd19cd7920d/src/COWShedResolver.sol#L52-L62

https://github.com/ensdomains/ens-contracts/blob/8e8cf71bc50fb1a5055dcf3d523d2ed54e725d28/contracts/registry/ENSRegistry.sol#L87-L96

Recommendation

Make sure that the entity is trusted, or do not revert when calling `ENS.setSubnodeRecord()` fails on proxy deployment with a `try / catch` block.

Response

The issue has been fixed by using a `try / catch` block instead of reverting on failed `ENS.setSubnodeRecord()` calls in commit https://github.com/cowdao-grants/cow-shed/commit/63eeff3c586ba6220a2707c20542cac7d019e97a

▼ [Low] Receive function in proxy does not fallback

Summary

The `COWShedProxy` implements a receive function that prevents delegating to the receive function of the implementation address.

Vulnerability Detail

The proxy implements a receive function:

```
contract COWShedProxy is COWShedStorage, Proxy {
        ...
    receive() external payable { }
    ...
}
```

Impact

If the proxy is used to delegate to an implementation with a receive function, this function will not be callable.

Code Snippets

https://github.com/cowdao-grants/cow-shed/blob/853206cd8ec43efbddda1bec7f2adfd19cd7920d/src/COWShedProxy.sol#L57

Recommendation

Remove this function. OpenZeppelin's proxy implementation correctly calls `_fallback()` in its payable fallback function: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/337bfd5ea4df9f7ebc755cd3cb4ecb3bd3d33fc7/contracts/proxy/Proxy.sol#L66

Response

The issue has been fixed by calling `_fallback()` in the receive function: https://github.com/cowdao-grants/cow-shed/commit/63eeff3c586ba6220a2707c20542cac7d019e97a.

▼ [Info] COWShedResolver does not implement supportsInterface()

Summary

The `COWShedResolver` implements certain functionality of an ENS resolver but does not implement `supportsInterface()` to inform users about supported functionality.

Vulnerability Detail

According to ENS' documentation, the resolver should follow EIP165 and return true on calls of `supportsInterface()` with supported interface values. The supported interfaces are:

- addr(bytes32 node) view returns (address) (`0x3b3b57de`)
- name(bytes32 node) view returns (string memory) (`0x691f3431`)

See https://docs.ens.domains/resolvers/interfaces

Impact

Users are misinformed about supported functionalities of the resolver.

Code Snippets

https://github.com/cowdao-grants/cow-shed/blob/853206cd8ec43efbddda1bec7f2adfd19cd7920d/src/COWShedResolver.sol

Recommendation

Implement that EIP165 `supportsInterface()` function.

Response

This issue has been fixed by correctly implementing `supportsInterface()` : https://github.com/cowdao-grants/cow-shed/commit/63eeff3c586ba6220a2707c20542cac7d019e97a