Base usernames

Security Review

Cantina Managed review by:
Christoph Michel, Lead Security Researcher
Akshay Srivastav, Associate Security Researcher
Rustyrabbit, Security Researcher

August 9, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Base is a secure and low-cost Ethereum layer-2 solution built to scale the userbase on-chain.

From Jul 17th to Jul 25th the Cantina team conducted a review of base-usernames on commit hash f889bb52. The team identified a total of **12** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 11

3 Findings

3.1 Low Risk

3.1.1 `RegistrarController` will use wrong premium at `launchTime`

Severity: Low Risk

Context: RegistrarController.sol#L386

Description: For previously registered names, the `RegistrarController` computes the premium based on the name's earliest available registration date (`expiry + GRACE_PERIOD`).

If a name has not been registered yet, it computes the premium based on `launchTime + GRACE_PERIOD` as `_getExpiry` returns `launchTime` and the `ExponentialPremiumPriceOracle` adds `GRACE_PERIOD`. The price oracle returns `0` if this date is not reached yet; the premium will be 0 from the `launchDate` up to `launchDate + GRACE_PERIOD`. Only afterwards, the real premium curve starts.

```
function _premium(string memory, uint256 expires, uint256) internal view override returns (uint256) {
    expires = expires + GRACE_PERIOD;
    if (expires > block.timestamp) {
        return 0;
    }
    uint256 elapsed = block.timestamp - expires;
    uint256 premium = decayedPremium(elapsed);
    if (premium > endValue) {
        return premium - endValue;
    }
    return 0;
}
```

The intended behavior is:

> Newly registered names will be priced with a premium based on the difference between the `block.timestamp` of the reigster call and the stored `launchTime`.

However, users can purchase new names with a premium of `0` at launch for up to `GRACE_PERIOD`.

Recommendation: For unregistered names, the premium should be computed based on `launchTime` alone, not `launchTime + GRACE_PERIOD`.

Consider rewriting the price oracle interface to take a final `expiry` timestamp that already includes any potential `GRACE_PERIOD`. Then add the `GRACE_PERIOD` in `_getExpiry` for already registered names, otherwise return `launchTime` without the grace period. Add tests for this scenario.

Coinbase: Fixed in PR 73.

Cantina Managed: Looks good, with the stipulation you should make sure `launchTime` never gets set in the future.

3.2 Informational

3.2.1 Null value is set as the resolver of reverse records for `EARegistrarController`

Severity: Informational

Context: EARegistrarController.sol#L284

Description: The `constructor` of `EARegistrarController` performs the `IReverseRegistrar.claim` call to claim its reverse records.

By analysing the deployment sequence of protocol contracts and the provided integration tests it can be seen that during the deployment of `EARegistrarController` the `ReverseRegistrar::defaultResolver` state variable is `address(0)` (as the default `L2Resolver` is not deployed yet). Hence the resolver of reverse records for `EARegistrarController` is set to `address(0)`.

Assuming that the `RegistrarController` will be deployed after early registration period, this issue should not impact `RegistrarController`.

Recommendation: Make sure the `owner` of `EARegistrarController` manually sets the resolver of reverse records for `EARegistrarController` after protocol deployment.

Proof of concept: This test was added to `test/IntegrationTest.t.sol`:

```
function test_emptyResolverForRegistrarController() public view {
    assertEq(registry.resolver(reverseRegistrar.node(address(registrarController))), address(0));
    assertEq(registry.resolver(reverseRegistrar.baseNode(address(registrarController))), address(0));
}
```

Coinbase: Fixed in PR 78.

Cantina Managed: Verified. Coinbase has decided to opt out of claiming reverse record of `EARegistrarController`.

3.2.2 Missing explicitly claiming the reverse resolution record for `L2Resolver`

Severity: Informational

Context: L2Resolver.sol#L110-L115

Description: The current implementation of `L2Resolver` contract does not claim the reverse resolution record for itself explicitly.

Ideally just like RegistrarController, EARegistrarController & ENS's PublicResolver reverse record should be explicitly claimed for `L2Resolver` during its construction.

Recommendation: Add the `IReverseRegistrar.claim` call in `L2Resolver::constructor`:

```
  constructor(ENS ens_, address registrarController_, address reverseRegistrar_, address owner_) {
      ens = ens_;
      registrarController = registrarController_;
      reverseRegistrar = reverseRegistrar_;
      _initializeOwner(owner_);
+     IReverseRegistrar(reverseRegistrar).claim(owner_);
  }
```

Coinbase: Fixed in PR 81.

Cantina Managed: Verified. The suggestion given by review team has been implemented.

3.2.3 `L2Resolver` does not indicate support for wildcard resolution

Severity: Informational

Context: L2Resolver.sol#L205-L222

Description: The `L2Resolver` does not include the interface for the extended resolver (`resolve()`) in the `supportsInterface()` function. Although ENSIP-19 specifies the resolver should always be called via the CCIP-read gateway clients can have their own use cases where this call is instantiated from the client for other purposes than ENSIP-19.

Recommendation: Add the extender resolver interface to the supported interfaces. Note that this isn't done in the `ExtendedResolver` from the ENS library itself.

Coinbase: Fixed in PR 77.

Cantina Managed: Verified.

3.2.4 Discounted registrants data can be lost when changing the registrar controller

Severity: Informational

Context: EARegistrarController.sol#L443, RegistrarController.sol#L469

Description: `discountedRegistrants`is being tracked in each of the registrar controllers separately. If the registrar controller is changed or replaced this information will be lost and users can claim the discount multiple times. This also applies when multiple controllers have the same discount active.

Recommendation: If replacing controller is a common use case consider keeping track of this information in a separate contract (possibly a discount validator router). If the discount is to be tracked per discount type this could also be done in the discount validators themselves.

Coinbase: Acknowledged. We expect very few early-access registrants and are not concerned with the possibility of ea participants getting a second discount when the GA launch goes live. For future registrar controllers, we can add some `legacyDiscountedRegistrants` check which checks the state of the existing `RegistrarController`.

Cantin Managed: Acknowledged.

3.2.5 Wildcard resolver not set when changing the default resolver on the reverse registrar

Severity: Informational

Context: ReverseRegistrar.sol#L115-L119

Description: When setting or changing the default resolver the resolver of the `[coinTypeAsHex].reverse` and `addr.reverse` nodes are not actually set, only the default resolver used by the `claim` and `setName` functions is changed.

ENSIP-19 specifies wildcard resolution should be supported:

> The Offchain resolver will also support wildcard of all the address subdomains with the format [address].[coinTypeAsHex].reverse.

Although this is primarily meant for the resolver on L1 it is recommended to also set this on L2.

Recommendation: Set the resolver for the `[coinTypeAsHex].reverse` and `addr.reverse` nodes in the `setDefaultResolver` so the wildcard resolver for reverse registrar follows the latest version of the default resolver.

Coinbase: Fixed in PR 76.

Cantina Managed: Verified.

3.2.6 L2 Registry contracts should never directly be used for resolutions

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: ENSIP-11 with ERC-3668 describe how cross-chain ENS forward resolutions can be implemented, ENSIP-19 describes how cross-chain reverse resolutions work.

Note that both resolutions always originate on the L1 chain by querying the `ENSRegistry` for a resolver for a specific node record. The L1 `ENSRegistry` acts as a starting point and the source of truth.

Recommendation: ENS resolutions cannot be achieved solely on L2 and contracts should refrain from attempting resolutions by directly interacting with only the L2 contracts.

Coinbase: Acknowledged. This is something we can address in our documentation since it's not possible to enforce on-chain.

Cantina Managed: Acknowledged.

3.2.7 Unclear why `addr.reverse` record is needed on L2

Severity: Informational

Context: ReverseRegistrar.sol#L153

Description: ENSIP-19 describes how cross-chain reverse resolution works. It looks for the primary name of the address on the wallet's currently connected network.

Therefore, if the network is ETH, it is backward-compatible with the standard reverse resolution described in ENSIP-3 using the resolver for the `[address].addr.reverse` record. (It also describes a forward resolution with the resolved primary name which could now be managed on an L2).

If the connected network is an L2, it will check the `[address].[coinTypeAsHex].reverse` instead. In Base's case, `80002105.reverse` is controlled by Base and will resolve to their `L1Resolver` which will use an EIP-10 off-chain lookup to query its L2 registry setup to resolve the `name(node)`.

In both cases, the `*.addr.reverse` records on L2's `ReverseRegistrar`/`BaseRegistrar` will not be read. It's unclear why they are needed and being claimed.

> Also note that the `RegistrarController`s only claim the base reverse records.

Recommendation: Clarify why `[address].addr.reverse` records are being claimed on L2.

Coinbase: Fixed in PR 74.

Cantina Managed: Verified. `[address].addr.reverse` records claiming functionality have been removed.

3.2.8 Missing state visibility specifiers

Severity: Informational

Context: Registry.sol#L29-L33

Description: The state of `Registry` is not annotated with visibility specifiers. The default visibility for state is `internal`.

Recommendation: Consider explicitly specifying the visibility as a best practice.

Coinbase: Fixed in PR 75.

Cantina Managed: Verified.

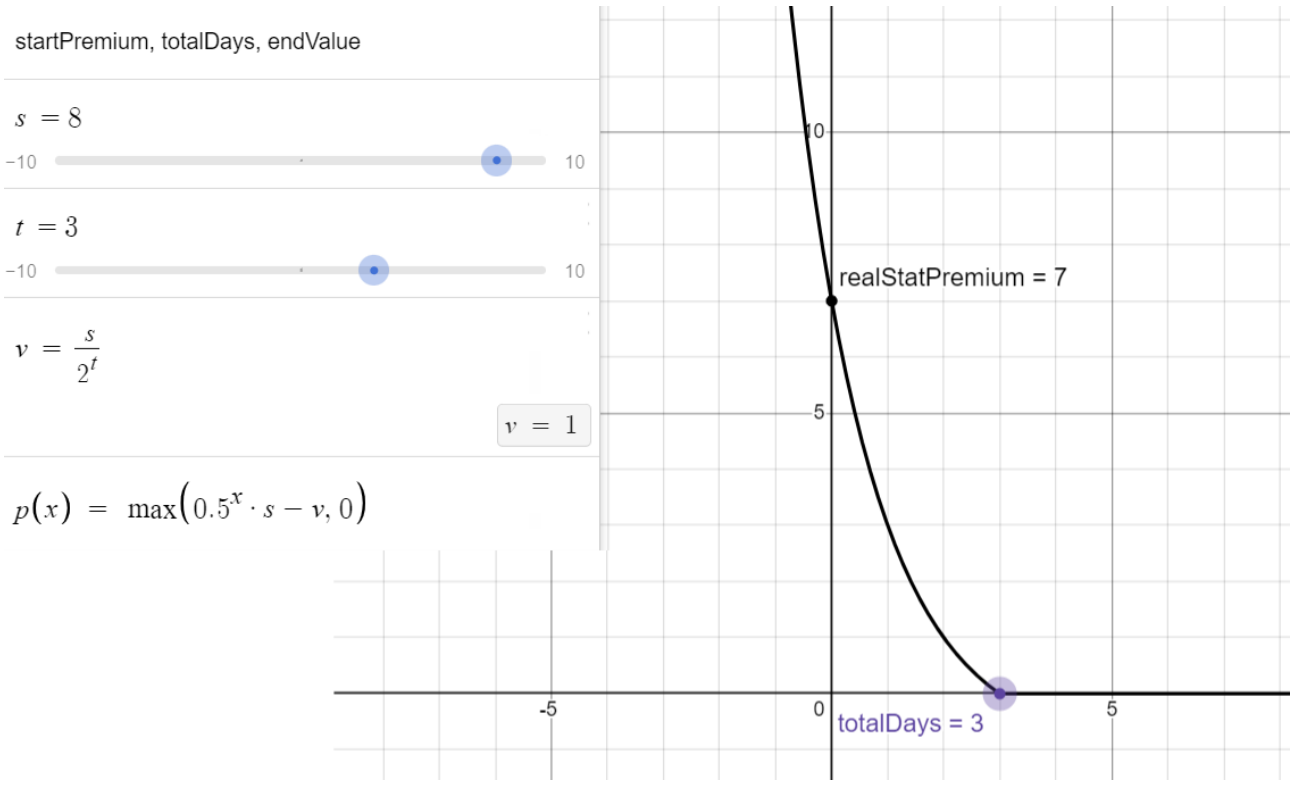
3.2.9 `ExponentialPremiumPriceOracle` start price clarifications

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The `ExponentialPremiumPriceOracle`'s premium is shifted by `endvalue` (depending on `totalDays`) such that after `totalDays` the premium is 0. However, that means the `startPremium` variable does not represent the actual start premium; the real start premium is the `startPremium` minus the `endvalue`. The premium curve will not have the exact exponential decay property of halving the premium after each day anymore (it decays faster than that).

See the illustration of the curve below:



Recommendation: Ensure the `startPremium` parameter is chosen such that the real desired start premium is `desiredStartPremium = startPremium - startPremium >> totalDays`, i.e. choose:

$$\text{startPremium} = \frac{\text{desiredStartPremium}}{1 - 2^{-\text{totalDays}}}$$

Coinbase: Acknowledged that as written it's `startPremium` is not precise. But given the decay rate at the beginning of the auction, I think the juice for making this change isn't worth the squeeze.

Cantina Managed: Acknowledged.

3.2.10 Typos & Documentation

Severity: Informational

Context: EARegistrarController.sol#L189, EARegistrarController.sol#L326, IPriceOracle.sol#L13, L2Resolver.sol#L176, L2Resolver.sol#L52, RegistrarController.sol#L336, ReverseRegistrar.sol#L135, ReverseRegistrar.sol#L204

Description: Consider fixing the following typos and documentation issues.

1. EARegistrarController.sol#L189: `PayemntReceiverUpdated` → `PaymentReceiverUpdated`.
2. L2Resolver.sol#L52, L2Resolver.sol#L176, ReverseRegistrar.sol#L204: `authroized` → `authorized`.
3. ReverseRegistrar.sol#L135: "The ENS node hash of the reverse record." The `claim` function claims two reverse records, it's unclear from the description which one it returns. Either return both nodes or clarify which node it returns.
4. IPriceOracle.sol#L13: `0` → `launchTime`.
5. RegistrarController.sol#L336 & EARegistrarController.sol#L326: Incorrect comment, `ReverseRegistrarUpdated` → `PaymentReceiverUpdated`

Coinbase: Fixed in PR 72.

Cantina Managed: Verified. The ReverseRegistrar.sol#L135 comment has been fixed as part of the fix for the issue "Unclear why `addr.reverse` record is needed on L2".

3.2.11 Unnecessary `NameRenewed` event in `EARegistrarController`

Severity: Informational

Context: EARegistrarController.sol#L184

Description: The `NameRenewed` event is not necessary for the `EARegistrarController` as this controller cannot `renew`.

Recommendation: Consider removing this event.

Coinbase: Fixed in PR 71.

Cantina Managed: Verified.