CANTINA

Nitro Validator

Security Review

Cantina Managed review by:

0xWeiss, Security Researcher
Om Parikh, Security Researcher

January 29, 2025

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 Dec 13th to Dec 25th the Cantina team conducted a review of nitro-validator on commit hash 7879d9b6. The team identified a total of **12** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	4	4	0
Low Risk	3	3	0
Gas Optimizations	1	1	0
Informational	4	3	1
Total	**12**	**11**	**1**

3 Findings

3.1 Medium Risk

3.1.1 False negatives in ECDSA384 can lead to DoS

Severity: Medium Risk

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

Description: The current pinned version of `solarity/libs/crypto/ECDSA384.sol` is an incomplete implementation, where correct signature is being rejected in some cases due to how point arithmetic is being handled.

See following github issues for more details

- Issue 126: False negatives in ECDSA384.sol.
- Issue 125: Inner Double Scalar Mul edge case.

Recommendation: Update the library to latest version and ensure it matches with wycheproof test suite's output for all cases. See PR 124 and PR 127.

Coinbase: Fixed in PR 4.

Cantina Managed: Fixed.

3.1.2 Incomplete Handling of CBOR Types and Additional Information Values

Severity: Medium Risk

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

Description: The `elementAt` function in the `CborDecode` library is responsible for parsing and extracting CBOR elements from a given byte array. However, the implementation has several concerns:

- When a CBOR simple or float value (major type `0xe0`) is passed, the function will not correctly handle the additional information (`ai`), which can lead to a situation where the function will truncate and interpret value falsely or read beyond bounds leading to undefined behaviour.
- For additional information (`ai`) values in the range of `28` to `30`, the function does not implement any specific handling or error checking. If such a value is encountered, the function will default to handling it as if it were a value with a small `ai` (less than `24`) and `ai = 31`, potentially resulting in incorrect parsing or data corruption.

Recommendation:

```
function elementAt(bytes memory cbor, uint256 ix, uint8 expectedType, bool required)
    internal
    pure
    returns (CborElement)
{
    uint8 _type = uint8(cbor[ix] & 0xe0);
    uint8 ai = uint8(cbor[ix] & 0x1f);

    if (_type == 0xe0) {
        require(!required || ai != 22, "null value for required element");
        // @audit floats & simple should not be allowed
        // primitive type, retain the additional information
        return LibCborElement.toCborElement(_type | ai, ix + 1, 0);
    }

    require(_type == expectedType, "unexpected type");

    if (ai == 24) {
        return LibCborElement.toCborElement(_type, ix + 2, uint8(cbor[ix + 1]));
    } else if (ai == 25) {
        return LibCborElement.toCborElement(_type, ix + 3, cbor.readUint16(ix + 1));
    } else if (ai == 26) {
        return LibCborElement.toCborElement(_type, ix + 5, cbor.readUint32(ix + 1));
    } else if (ai == 27) {
        return LibCborElement.toCborElement(_type, ix + 9, cbor.readUint64(ix + 1));
    }

    // @audit should revert for ai >= 28 and ai <= 30

    // this is for ai = 31 and ai <= 23
    return LibCborElement.toCborElement(_type, ix + 1, ai);
}
```

Coinbase: Fixed in PR 5.

Cantina Managed: Fixed.

3.1.3 Missing validation checks in `timestampAt`

Severity: Medium Risk

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

Description: - The function does not verify the `ASN.1` tags (`0x17` for UTCTime and `0x18` for Generalized-Time), which is crucial for determining the correct timestamp type.

- The function incorrectly uses the length of the timestamp data to determine its type which is not reliable
- There is no validation to ensure the timestamp ends with 'Z' to indicate UTC, which is essential for accurate time conversion.
- The function fails to check if the bytes being converted to digits are valid (0-9), leading to potential misinterpretation errors.

Recommendation:

```
function timestampAt(bytes memory der, Asn1Ptr ptr) internal pure returns (uint256) {
    uint16 _years;

    uint256 offset = ptr.content();
    uint256 length = ptr.length();

    // @audit validate tag for UTCTime and GeneralizedTime with length check for each of them
    // @audit validate timezone is UTC

    if (length == 13) {
        _years = (uint8(der[offset]) - 48 < 5) ? 2000 : 1900;
    } else {
        // @audit check length is 15
        _years = (uint8(der[offset]) - 48) * 1000 + (uint8(der[offset + 1]) - 48) * 100;
        offset += 2;
    }

    _years += (uint8(der[offset]) - 48) * 10 + uint8(der[offset + 1]) - 48;

    // @audit subtracting 48 may not be reliable to convert to ASCII depending on use-case
    // der[offset + index] >= 48 and der[offset + index] <= 57  may be required here

    uint8 _months = (uint8(der[offset + 2]) - 48) * 10 + uint8(der[offset + 3]) - 48;
    uint8 _days = (uint8(der[offset + 4]) - 48) * 10 + uint8(der[offset + 5]) - 48;
    uint8 _hours = (uint8(der[offset + 6]) - 48) * 10 + uint8(der[offset + 7]) - 48;
    uint8 _mins = (uint8(der[offset + 8]) - 48) * 10 + uint8(der[offset + 9]) - 48;
    uint8 _secs = (uint8(der[offset + 10]) - 48) * 10 + uint8(der[offset + 11]) - 48;

    return timestampFromDateTime(_years, _months, _days, _hours, _mins, _secs);
}
```

Coinbase: Fixed in PR 6.

Cantina Managed: Fixed.

3.1.4 Missing out-of-bounds check

Severity: Medium Risk

Context: LibBytes.sol#L5

Description: A possible `out-of-bounds` error could arise from attempting to hash memory beyond the bounds of the bytes array.

```
function keccak(bytes memory data, uint256 offset, uint256 length) internal pure returns (bytes32 result) {
    assembly {
        result := keccak256(add(data, add(32, offset)), length)
    }
}
```

If the offset is greater than or equal to the length of the array, the pointer `add(data, add(32, offset))` will point outside the valid memory range. Therefore, the function will attempt to access memory outside the allocated bounds.

Recommendation: Add the following check validating `offset` and `length` against `data.length`:

```
  function keccak(bytes memory data, uint256 offset, uint256 length) internal pure returns (bytes32 result) {
+     require(offset + length <= data.length, "Out-of-bounds access");
      assembly {
          result := keccak256(add(data, add(32, offset)), length)
      }
  }
```

Coinbase: Fixed in PR 10.

Cantina Managed: Fixed.

3.2 Low Risk

3.2.1 Missing validation checks in `timestampFromDateTime`

Severity: Low Risk

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

Description: In function `timestampFromDateTime`, it checks if `year >= 1970` but other input fields are not checked. Any values can be passed for `month`, `day`, `hour`, `minute`, `second` and will be accepted leading to undefined behaviour.

This may be protected currently implicity by assuring that signed data was not tampered with at call site where this library is currently used but if `Asn1Decode` library is used in isloation, it will lead to issues described above.

Recommendation:

```
function timestampFromDateTime(
    uint256 year,
    uint256 month,
    uint256 day,
    uint256 hour,
    uint256 minute,
    uint256 second
) internal pure returns (uint256) {
    require(year >= 1970);

    // @audit add validator for month, day, hour, minute, second

    int256 _year = int256(year);
    int256 _month = int256(month);
    int256 _day = int256(day);

    int256 _days = _day - 32075 + 1461 * (_year + 4800 + (_month - 14) / 12) / 4
        + 367 * (_month - 2 - (_month - 14) / 12 * 12) / 12 - 3 * ((_year + 4900 + (_month - 14) / 12) / 100)
        ↪  / 4
        - 2440588;

    return ((uint256(_days) * 24 + hour) * 60 + minute) * 60 + second;
}
```

Coinbase: Fixed in PR 7.

Cantina Managed: Fixed.

3.2.2 Unsafe Truncation of `ASN.1 DER` Length Value to `uint80` Without Bounds Validation

Severity: Low Risk

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

Description: The `readNodeLength` function incorrectly handles ASN.1 DER length fields by truncating large length values to uint80 without validation. According to ASN.1 specification, the long-form length encoding can include up to 127 bytes to represent the length value, but the function silently truncates this to uint80 (max value $2^{80} - 1$) in:

```
return LibAsn1Ptr.toAsn1Ptr(ix, ixFirstContentByte, uint80(length));
```

However, in practice, such large data might not encountered due to constraints.

Recommendation:

```
function readNodeLength(bytes memory der, uint256 ix) private pure returns (Asn1Ptr) {
    uint256 length;
    uint80 ixFirstContentByte;

    if ((der[ix + 1] & 0x80) == 0) {
        length = uint8(der[ix + 1]);
        ixFirstContentByte = uint80(ix + 2);
    } else {
        uint8 lengthbytesLength = uint8(der[ix + 1] & 0x7F);

        if (lengthbytesLength == 1) {
            length = uint8(der[ix + 2]);
        } else if (lengthbytesLength == 2) {
            length = der.readUint16(ix + 2);
        } else {
            length = uint256(readBytesN(der, ix + 2, lengthbytesLength) >> (32 - lengthbytesLength) * 8);
        }

        ixFirstContentByte = uint80(ix + 2 + lengthbytesLength);
    }

    // @audit max len as per spec is reading 127 bytes, so 2**127 - 1 , uint80 is truncating it without length
    ↪  check
    // change to throw if length is more than 80 bytes
    return LibAsn1Ptr.toAsn1Ptr(ix, ixFirstContentByte, uint80(length));
}
```

Coinbase: Fixed in PR 9.

Cantina Managed: Fixed.

3.2.3 An arbitrary length can be specified in different functions

Severity: Low Risk

Context: Sha2Ext.sol#L11, Sha2Ext.sol#L136, Sha2Ext.sol#L151

Description: Different hashing functions allow to specify a length as a parameter:

```
function sha2(bytes memory message, uint256 offset, uint256 length, uint64[8] memory h) internal pure {

function sha384(bytes memory message, uint256 offset, uint256 length) internal pure returns (bytes memory)
```

This length is meant to be the length of the bytes array but it is not specified allowing for an incorrect length to be passed.

Recommendation: Do use the `message.length` value instead of allowing to arbitrarily pass the length as a parameter.

Coinbase: Fixed in PR 14. The offset + length parameters are required as we are taking the `SHA384` of a subset of the message. We added bounds checks to it instead.

Cantina Managed: Fixed.

3.3 Gas Optimization

3.3.1 Hardcoding keys would cost less gas on deployment

Severity: Gas Optimization

Context: NitroValidator.sol#L24

Description: The variable declaration of the following keys can be cheaper if they were hardcoded:

```
bytes32 public constant ATTESTATION_TBS_PREFIX = keccak256(hex"846a5369676e61747572653144a101382240");
bytes32 public constant ATTESTATION_DIGEST = keccak256("SHA384");
bytes32 public constant CERTIFICATE_KEY = keccak256(bytes("certificate"));
bytes32 public constant PUBLIC_KEY_KEY = keccak256(bytes("public_key"));
bytes32 public constant MODULE_ID_KEY = keccak256(bytes("module_id"));
bytes32 public constant TIMESTAMP_KEY = keccak256(bytes("timestamp"));
bytes32 public constant USER_DATA_KEY = keccak256(bytes("user_data"));
bytes32 public constant CABUNDLE_KEY = keccak256(bytes("cabundle"));
bytes32 public constant DIGEST_KEY = keccak256(bytes("digest"));
bytes32 public constant NONCE_KEY = keccak256(bytes("nonce"));
bytes32 public constant PCRS_KEY = keccak256(bytes("pcrs"));
```

Recommendation: Hardcode the following keys:

```
- bytes32 public constant ATTESTATION_TBS_PREFIX = keccak256(hex"846a5369676e61747572653144a101382240");
- bytes32 public constant ATTESTATION_DIGEST = keccak256("SHA384");
- bytes32 public constant CERTIFICATE_KEY = keccak256(bytes("certificate"));
- bytes32 public constant PUBLIC_KEY_KEY = keccak256(bytes("public_key"));
- bytes32 public constant MODULE_ID_KEY = keccak256(bytes("module_id"));
- bytes32 public constant TIMESTAMP_KEY = keccak256(bytes("timestamp"));
- bytes32 public constant USER_DATA_KEY = keccak256(bytes("user_data"));
- bytes32 public constant CABUNDLE_KEY = keccak256(bytes("cabundle"));
- bytes32 public constant DIGEST_KEY = keccak256(bytes("digest"));
- bytes32 public constant NONCE_KEY = keccak256(bytes("nonce"));
- bytes32 public constant PCRS_KEY = keccak256(bytes("pcrs"));

+ bytes32 public constant ATTESTATION_TBS_PREFIX =
↪  0x63ce814bd924c1ef12c43686e4cbf48ed1639a78387b0570c23ca921e8ce071c;
+ bytes32 public constant ATTESTATION_DIGEST =
↪  0x501a3a7a4e0cf54b03f2488098bdd59bc1c2e8d741a300d6b25926d531733fef;
+ bytes32 public constant CERTIFICATE_KEY = 0x925cec779426f44d8d555e01d2683a3a765ce2fa7562ca7352aeb09dfc57ea6a;
+ bytes32 public constant PUBLIC_KEY_KEY = 0xc7b28019ccfdbd30ffc65951d94bb85c9e2b8434111a000b5afd533ce65f57a4;
+ bytes32 public constant MODULE_ID_KEY = 0x8ce577cf664c36ba5130242bf5790c2675e9f4e6986a842b607821bee25372ee;
+ bytes32 public constant TIMESTAMP_KEY = 0x4ebf727c48eac2c66272456b06a885c5cc03e54d140f63b63b6fd10c1227958e;
+ bytes32 public constant USER_DATA_KEY = 0x5e4ea5393e4327b3014bc32f2264336b0d1ee84a4cfd197c8ad7e1e16829a16a;
+ bytes32 public constant CABUNDLE_KEY = 0x8a8cb7aa1da17ada103546ae6b4e13ccc2fafa17adf5f93925e0a0a4e5681a6a;
+ bytes32 public constant DIGEST_KEY = 0x682a7e258d80bd2421d3103cbe71e3e3b82138116756b97b8256f061dc2f11fb;
+ bytes32 public constant NONCE_KEY = 0x7ab1577440dd7bedf920cb6de2f9fc6bf7ba98c78c85a3fa1f8311aac95e1759;
+ bytes32 public constant PCRS_KEY = 0x61585f8bc67a4b6d5891a4639a074964ac66fc2241dc0b36c157dc101325367a;
```

Coinbase: Fixed in PR 2.

Cantina Managed: Fixed.

3.4 Informational

3.4.1 Unused code

Severity: Informational

Context: NitroValidator.sol#L7, NitroValidator.sol#L12

Description: The following imports are unused across the repository:

- `import {Asn1Decode} from "./Asn1Decode.sol";` from `NitroValidator.sol`.
- `import {console} from "forge-std/console.sol";` from `NitroValidator.sol`.
- `import {Sha2Ext} from "./Sha2Ext.sol";` from from `ICertManager.sol`.
- `import {Asn1Decode, Asn1Ptr, LibAsn1Ptr} from "./Asn1Decode.sol";` from `ICertManager.sol`.
- `import {LibBytes} from "./LibBytes.sol";` from `ICertManager.sol`.

Recommendation: Remove unused code.

Coinbase: Fixed in PR 3.

Cantina Managed: Fixed.

3.4.2 `readNodeLength` assumes `ASN1` tag is always 1-byte

Severity: Informational

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

Description: In `readNodeLength`, it reads and checks first-byte as tag:

```
if ((der[ix + 1] & 0x80) == 0)
```

Which is not a spec-compliant implementation. As per Section 8.1.2.4.2 of ITU-T X.690, tags can be encoded with more than 1-byte. however, in x509 certificates parsing, all tags are 1-byte.

Recommendation: Document that library is only comptaible decoding ASN1 DER data where tag is 1-byte long.

Coinbase: Fixed in PR 8.

Cantina Managed: Fixed.

3.4.3 Missing event when is certificate is verified

Severity: Informational

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

Description:

```
mapping(bytes32 => bytes) public verified;
```

In `CertManager` when a new certifcate hash is added, it doesn't emit event so there is no way to track and observe offchain.

Recommendation: emit an event with certificate hash and other necessarty contents

Coinbase: Fixed in PR 11.

Cantina Managed: Fixed.

3.4.4 Lack of support for certificate revokation

Severity: Informational

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

Description: The CertManager contract does not support certificate revocation, preventing the invalidation of compromised certificates or CAs before their expiration date. This could allow compromised certificates to remain trusted even after they have been reported as compromised.

A compromised intermediate CA certificate could lead to acceptance of unauthorized attestations in the Nitro protocol until certificate expiration.

Recommendation: Consider documenting the risk or adding support for revokation

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.