Veridise

Auditing Report

Hardening Blockchain Security with Formal Methods

FOR

Aleo Oracle Program

Veridise Inc.

August 07, 2025

► **Prepared For:**

Venture23 Inc.
https://www.venture23.xyz/

► **Prepared By:**

Mark Anthony
Alberto Gonzalez

► **Contact Us:**

contact@veridise.com

► **Version History:**

Sep. 25, 2025 V2
Aug. 18, 2025 V1

Contents

1 Executive Summary

From Aug. 7, 2025 to Aug. 11, 2025, Venture23 Inc. engaged Veridise to conduct a security assessment of their Aleo Oracle Program. The security assessment covered the Leo smart contract components of the Aleo Oracle protocol. Veridise conducted the assessment over 6 person-days, with 2 security analysts reviewing the project over 3 days on commit `d709d0e`. The review strategy involved a thorough, manual code review of the program source code performed by Veridise security analysts. This security assessment was performed concurrently with the notarization and verification backend components of the Aleo Oracle protocol*.

Project Summary. The Oracle Leo Program is the on-chain component of the protocol that stores TEE-attested pricing data produced by the backend in response to notarization requests. Each submission includes a TEE report and a signature from a whitelisted Aleo address corresponding to a key pair generated inside the TEE; the backend currently supports Intel SGX and AWS Nitro. On-chain, the program verifies that the signature is valid and that the report's enclave identity matches the configured reference (SGX unique_id or Nitro PCR values). The program does not verify on-chain that the report was generated by a genuine TEE as remote verification of vendor attestations is not implemented. Accordingly, it assumes the key pair used to derive the whitelisted address was created inside the TEE and that the private key lives only inside the enclave; under this assumption, a matching identity check in the on-chain program indicates the data originated from the intended enclave running a genuine TEE. Only when these checks succeed is the price recorded on-chain for consumption by other Aleo programs. Notarization requests are not limited to just pricing data, and HTTP requests can be made to any domain as long as the domain is whitelisted in the notarization backend.

Code Assessment. The Aleo Oracle Program developers provided the source code of the Aleo Oracle Program for the code review. The source code was initially created by a separate development team and was subsequently extended and modified by Venture23. It contains some documentation in the form of documentation comments on functions and storage variables. To facilitate the Veridise security analysts' understanding of the code, the Aleo Oracle Program developers shared documentation for the Oracle Program (†), and met the Veridise analysts to provide an initial walkthrough of the codebase.

The source code contained a test suite, which the Veridise security analysts noted adequately covered the core functional flows as well as relevant negative test cases.

Summary of Issues Detected. The security assessment uncovered 8 issues, 1 of which is assessed to be of a low severity by the Veridise analysts. Specifically, V-AOP-VUL-001 describes how the oracle program hardcodes the owner address which prevents ownership transfer and can cause operational issues if migration is required. The Veridise analysts also identified 6

* The audit report for these components can be found on Veridise's website at `https://veridise.com/audits-archive/`

† Documentation for the Oracle program is available at `https://aleo-oracle-docs.surge.sh/guide/`

warning-severity issues including V-AOP-VUL-002 which explains how the timestamped hash used to perform historical requests does not contain the enclave configuration which may cause historical queries to return valid attestations from unexpected enclave configurations, V-AOP-VUL-004 that describes how the oracle program accepts attested data without checking the HTTP response status code, which may allow empty or partial responses to be stored as valid updates, and V-AOP-VUL-006 that notes the absence of a mechanism to revoke compromised attested data. Additionally, the security review also uncovered 1 informational finding.

Recommendations. After conducting the assessment of the protocol, the security analysts had a few suggestions to improve the Aleo Oracle Program security.

Implement an extensive pausability framework. The analysts recommend implementing an extensive pausability framework with both centralized and decentralized options. A pause action governed by a contract or an admin pause should be implemented as an immediate safeguard, allowing trusted parties to halt critical operations at the first sign of trouble. Alongside this, consider implementing a permissionless "pause with proof" mechanism, allowing any participant to submit verifiable evidence of an invalid quote and automatically suspend further state updates if the on-chain verification of the invalid quote fails. As further discussed in V-AOP-VUL-003, this dual approach ensures rapid containment through centralized operations while enabling the community to intervene in provable emergencies, reducing reliance on a single point of control and limiting the damage in case of compromised oracle data.

Domain separation for allowed public keys. The analysts recommend maintaining distinct mappings for allowed public keys associated with the Intel SGX and AWS Nitro environments. Using a single mapping for both enclave types risks accidental key overlap, misconfiguration, or incorrect attestation acceptance if a key intended for one environment is mistakenly used in the other. By enforcing domain separation at the storage level, the oracle program can ensure that only the correct enclave type is matched against the corresponding set of authorized signing keys.

2 Project Dashboard

Table 2.1: Application Summary.

Name	Version	Type	Platform
Aleo Oracle Program	`d709d0e`	Leo	Aleo

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Aug. 7–Aug. 11, 2025	Manual	2	6 person-days

Table 2.3: Vulnerability Summary.

Name	Number	Acknowledged	Fixed
Critical-Severity Issues	0	0	0
High-Severity Issues	0	0	0
Medium-Severity Issues	0	0	0
Low-Severity Issues	1	1	1
Warning-Severity Issues	6	6	2
Informational-Severity Issues	1	1	1
TOTAL	8	8	4

Table 2.4: Category Breakdown.

Name	Number
Maintainability	3
Data Validation	3
Usability Issue	1
Authorization	1

3 Security Assessment Goals and Scope

3.1 Security Assessment Goals

The engagement was scoped to provide a security assessment of Aleo Oracle Program's source code. During the assessment, the security analysts aimed to answer questions such as:

- Are there any access control issues in the program?
- Are the enclave identity verification checks implemented correctly and in accordance with vendor documentation?
- Does the program implement a pause mechanism in case the report-signing private key is compromised?
- Can a malicious user exploit the replay of previously submitted prices?
- Is the program tracking the most up-to-date price correctly?
- Can `request_hash` be reused across distinct request types, causing unintended price overwrites?
- Is the program vulnerable to common pitfalls specific to Leo programs?
- Does the oracle make provisions to revoke or invalidate attested data associated with compromised or hitherto unsupported enclave configurations?

3.2 Security Assessment Methodology & Scope

Security Assessment Methodology. To address the questions above, the security assessment involved a thorough manual review of the Aleo program source code conducted by human experts.

Scope. The scope of this security assessment is limited to `programs/vlink_oracle_v0001.leo` file as agreed upon with the Aleo Oracle Program developers.

Methodology. Veridise security analysts inspected the provided tests, and read the provided Aleo Oracle Program documentation. They then began a manual review of the code.

Before the security assessment, the Veridise security analysts met with the Aleo Oracle Program developers to ask questions about the code, and get a high level overview of the protocol design.

3.3 Classification of Vulnerabilities

When Veridise security analysts discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise.

The severity of a vulnerability is evaluated according to the Table 3.1.

The likelihood of a vulnerability is evaluated according to the Table 3.2.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

Table 3.2: Likelihood Breakdown

Not Likely	A small set of users must make a specific mistake
Likely	Requires a complex series of steps by almost any user(s) - OR - Requires a small set of users to perform an action
Very Likely	Can be easily performed by almost anyone

The impact of a vulnerability is evaluated according to the Table 3.3:

Table 3.3: Impact Breakdown

Somewhat Bad	Inconveniences a small number of users and can be fixed by the user
Bad	Affects a large number of people and can be fixed by the user - OR - Affects a very small number of people and requires aid to fix
Very Bad	Affects a large number of people and requires aid to fix - OR - Disrupts the intended behavior of the protocol for a small group of users through no fault of their own
Protocol Breaking	Disrupts the intended behavior of the protocol for a large group of users through no fault of their own

4 Trust Model

4.1 Operational Assumptions.

In addition to assuming that any out-of-scope components behave correctly, Veridise analysts assumed the following properties held when modeling security for the Aleo Oracle Program.

- **Encoding of Attestation Requests:** Prices are stored in the program using the hash of the encoded attestation request as the key. It is assumed that the request encoding scheme guarantees uniqueness, meaning that distinct requests always produce the same encoding and therefore deterministic hashes, meaning that identical requests always produce the same encoding.
- **Verification for Stale Prices:** The Aleo program determines the latest price by comparing the timestamps attached to each report. If the timestamp of an incoming price is greater than the currently stored latest timestamp, the new price replaces the previous one as the latest. This relies on the assumption that the timestamp embedded in the TEE report, together with the attested price, has been validated by the backend to reflect the time at which the price was obtained from the external data source as accurately as possible. If the source returns a stale price (for example, CoinGecko includes a `last_updated_at` field in its response*) and the backend notarizes it using its local clock rather than the source's freshness time, the program will treat that stale value as newer, overwrite a fresher price, and cause downstream consumers to read outdated data as the latest price.
- **No private key leakage:** A report is accepted as authentic if it carries a valid signature from a whitelisted Aleo address. The program does not verify that attested reports originate from genuine TEEs. Therefore, security relies on the off-chain guarantee that the registered public key corresponds to a key pair generated inside the TEE and that the private key has not been leaked or used outside the enclave. If the private key is compromised, an attacker can submit arbitrary prices that satisfy the on-chain validations.

4.2 Privileged Roles.

Roles. This section describes in detail the specific roles present in the system, and the actions each role is trusted to perform. *Privileged* roles like an administrator or owner may have a critical impact on the protocol if compromised.

During the review, Veridise analysts assume that the role operators perform their responsibilities as intended. Protocol exploits relying on the below roles acting outside of their privileged scope are considered outside of scope.

- *Owner.* The owner of the Oracle program can configure the Intel SGX unique_id and the AWS Nitro PCR values used to verify the enclave configurations on-chain. The owner also maintains the allowlist of Aleo public keys trusted by the Oracle program keys, that the

* See `https://docs.coingecko.com/reference/simple-price`

notarization backend uses to sign attested data. A compromised owner could fully rewrite the Oracle's trust anchors, authorizing their own signing key, setting enclave identifiers and PCR values to match arbitrary reports, and submitting data that passes the contract's internal validations. At present, the owner is defined as a hard-coded constant in the Oracle program, which leaves no possibility for migration or transfer of ownership. This design choice is risky in a rapidly evolving ecosystem like Aleo, as it prevents adapting to governance changes or any required ownership changes. V-AOP-VUL-001 describes the risks related to having a constant owner in further detail.

Operational Recommendations. Highly-privileged operations should be operated by a multi-sig contract or decentralized governance system. These operations should be guarded by a timelock to ensure there is enough time for incident response. In case of decentralized governance, such operations should be tested in example scenarios to ensure the council members are available and ready to respond when necessary.

Full validation of operational security practices is beyond the scope of this review. Users of the protocol should ensure they are confident that the operators of privileged keys are following best practices such as:

- Never storing a protocol key in plaintext, on a regularly used phone, laptop, or device, or relying on a custom solution for key management.
- Using separate keys for each separate function.
- Storing multi-sig keys in a diverse set of key management software/hardware services and geographic locations.
- Enabling 2FA for key management accounts. SMS should *not* be used for 2FA, nor should any account which uses SMS for 2FA. Authentication apps or hardware are preferred.
- Validating that no party has control over multiple multi-sig keys.
- Performing regularly scheduled key rotations for high-frequency operations.
- Securely storing physical, non-digital backups for critical keys.
- Actively monitoring for unexpected invocation of critical operations and/or deployed attack contracts.
- Regularly drilling responses to situations requiring emergency response such as pausing/unpausing.

5 Vulnerability Report

This section presents the vulnerabilities found during the security assessment. For each issue found, the type of the issue, its severity, location in the code base, and its current status (i.e., acknowledged, fixed, etc.) is specified. Table 5.1 summarizes the issues discovered:

Table 5.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-AOP-VUL-001	Non-transferrable owner address due to . . .	Low	Fixed
V-AOP-VUL-002	Historical lookup key does not include . . .	Warning	Acknowledged
V-AOP-VUL-003	Protocol lacks an emergency pause . . .	Warning	Partially Fixed
V-AOP-VUL-004	Missing verification of http response status	Warning	Fixed
V-AOP-VUL-005	Missing validation on attestation timestamp	Warning	Acknowledged
V-AOP-VUL-006	Lack of mechanism to revoke invalid or . . .	Warning	Fixed
V-AOP-VUL-007	Owner can set PCR values to zero enabling . . .	Warning	Acknowledged
V-AOP-VUL-008	Maintainability concerns	Info	Fixed

5.1 Detailed Description of Issues

5.1.1 V-AOP-VUL-001: Non-transferrable owner address due to constant definition

Severity	Low	**Commit**	N/A
Type	Maintainability	**Status**	Fixed
Location(s)	`programs/vlink_oracle_v0001.leo:42`		
Confirmed Fix At	https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1, https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1, 2e0b44d, 81b97c5		

Description The oracle Aleo program defines the contract owner as a constant address, making it permanently fixed. This means the owner cannot be rotated even if the private key is compromised, lost, or if the administrative control needs to change, such as migrating ownership to a governance based council.

```
1  // owner of the oracle program
2  const owner: address =
     aleo1rhgdu77hgyqd3xjj8ucu3jj9r2krwz6mnzyd80gncr5fxcwlh5rsvzp9px;
```

Any ownership change would require deploying a new oracle program and reconfiguring every dependent on-chain program. This is not feasible at scale since it would be very costly in terms of effort and gas costs to migrate ownership of the oracle.

Impact The inability to change the owner address can be very troublesome in recovery or migration scenarios, causing operational issues and downtime for all programs that rely on the oracle.

Recommendation The owner should be stored in mutable state such as a mapping, allowing a controlled `transfer_ownership` procedure that can only be initiated by the current owner. Ideally, ownership should be held by a governance-controlled program such as a council, to avoid having a single source of failure.

Developer Response The developers replaced the constant `owner` address with an `admin` mapping and the needed functionality to assign the role to a new account.

5.1.2 V-AOP-VUL-002: Historical lookup key does not include enclave identity

Severity	Warning	**Commit**	N/A
Type	Usability Issue	**Status**	Acknowledged
Location(s)	`programs/vlink_oracle_v0001.leo:28-31`		
Confirmed Fix At	N/A		

Description The program defines a struct `TimestampedHash`, which can be used as a historical reference to a request hash and a particular attestation timestamp. But, this composite key does not bind the referenced value to the enclave identity or the configuration under which it was produced. See snippet below for reference.

```
struct TimestampedHash {
  request_hash: u128,
  attestation_timestamp: u128
}
```

If the trusted SGX `MRENCLAVE`(unique ID) or Nitro PCR set changes over time, two attestations for the same request and timestamp range can correspond to different trust roots. Users querying history may unknowingly retrieve data produced under an unintended enclave configuration, especially if they are unaware that the oracle rotated its enclave or updated PCR expectations.

Impact Queries on historical data can return values that have valid attestations, but are tied to a different enclave configuration than the caller expects. This can undermine the assumptions consumers make about the fetched oracle response. In extreme cases, integrators may make policy or financial decisions based on data that was produced under a configuration they do not expect.

Recommendation Bind historical references to the TEE configuration that produced the quote. Extend the `TimestampedHash` to include an enclave-configuration field digest that captures the SGX measurement/unique ID, or the relevant Nitro PCR values. Store this digest alongside `request_hash` and `attestation_timestamp`, and use this key to perform the historical requests.

Developer Response The developers have acknowledged the issue but indicated they do not plan to provide a fix:

"We won't be using historical lookup feature that often outside the contract. Also we will have only one sgx unique id at a time in the aleo program so we think it is not accessary include the sgx unique id in the timestamped hash."

5.1.3 V-AOP-VUL-003: Protocol lacks an emergency pause mechanism

Severity	Warning	**Commit**	N/A
Type	Authorization	**Status**	Partially Fixed
Location(s)	`programs/vlink_oracle_v0001.leo`		
Confirmed Fix At	`https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1`, 81b97c5		

Description The current oracle design lacks a mechanism to promptly pause critical operations during an emergency. If attestation validation is flawed, a trusted enclave is misconfigured, the admin key is compromised, or incorrect data begins propagating, there is no way to halt new submissions or temporarily disable critical oracle functions. As a result, external programs that depend on this oracle may continue ingesting compromised data, further amplifying the impact.

Even if a centralized pause mechanism were added, it would still depend on the owner detecting the problem and acting, which introduces delay. In high-risk scenarios, such as a leaked enclave signing key-reports may continue to verify at the application layer, and there is no permissionless, on-chain route for third parties to demonstrate the issue and stop further harm. Providing a "pause with proof" path would allow external observers who detect an invalid quote to supply evidence against the quote and trigger an immediate pause if the proof is successfully verified on-chain.

Impact Incorrect or forged reports may continue to be accepted and propagated after a compromise, causing dependent programs to consume corrupted data and make unsafe decisions.

Without a permissionless "pause with proof" path, external responders cannot help contain the incident with prompt response, even when they can present the original quote and demonstrate on-chain that verification fails. This will result in the damage spreading until the owner is alerted and intervenes.

Recommendation Introduce a dual layer pausing mechanism. First, add an owner or governance-gated pause that immediately disables state-changing transitions. Second, add a permissionless challenge that pauses the oracle if a challenger can present evidence that a previously committed quote is invalid. If the on-chain verification of the quote fails, then pause all state updates to the oracle.

To make this feasible without storing entire quotes, persist a compact commitment of the quote at submission time, which can be used to refer to the quote when providing evidence against it. Expose a `challenge` transition that accepts the full quote string, recomputes the commitment, and then runs the expensive ECDSA signature verification on-chain.

Developer Response The developers introduced functionality for the owner and only the owner to pause and unpause the protocol. This status is also validated to be unpaused when sgx/nitro reports are posted to the oracle.

However, a permissionless mechanism that pauses the oracle if a challenger can present evidence that a previously committed quote is invalid, is not implemented.

5.1.4 V-AOP-VUL-004: Missing verification of http response status

Severity	Warning	**Commit**	N/A
Type	Data Validation	**Status**	Fixed
Location(s)	`programs/vlink_oracle_v0001.leo`		
Confirmed Fix At	`https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1`, 20e9b0c		

Description The oracle program accepts report data without enforcing that the HTTP request to the attestation target succeeded. Although the Oracle data within the report includes a `responseStatusCode` field, the Aleo program does not check it before storing the attested data. This may allow updates derived from failed or redirected requests to be accepted as valid as long as the signature verification succeeds, which can result in empty or partial responses being stored as attested data.

Impact Error pages, empty bodies, or partial responses may be accepted as fresh oracle updates, which can cause issues for integrators and consumers that rely on the oracle for accurate data.

Recommendation Verify that the `responseStatusCode` within the oracle data is `200`, to ensure that the api request to the attestation target was successful.

Developer Response The developers now verify the `responseStatusCode` on-chain, as recommended.

5.1.5 V-AOP-VUL-005: Missing validation on attestation timestamp

Severity	Warning	**Commit**	N/A
Type	Data Validation	**Status**	Acknowledged
Location(s)	`programs/vlink_oracle_v0001.leo:371-373`		
Confirmed Fix At	N/A		

Description The oracle accepts attestations for a request hash through quotes that include an attestation timestamp, but there is no check performed to ensure the attestation's timestamp is **not in the future** relative to the current timestamp on-chain. While SGX should not be able to truthfully attest to a future time, clock skew, misconfiguration, or a compromised enclave can still produce a timestamp dated in the future. If the provided attestation timestamp is greater than the latest timestamp, the latest attested data is overwritten with the provided data. See snippet below for context.

```
1    if (attested_data.attestation_timestamp > latest_data.attestation_timestamp) {
2      Mapping::set(sgx_attested_data, request_hash, attested_data);
3    }
```

But, if the attestation timestamp is one far into the future, the latest attested data cannot be updated until the particular timestamp is crossed. This can be used by an attacker to cause a denial of service and prevent updates to the latest attested data.

Impact If an attestation lands with a timestamp far in the future, subsequent legitimate updates will be rejected as they will be viewed as stale updates and the oracle's "latest" value will remain frozen until the future timestamp value is reached.

Recommendation When accepting an attestation, verify that the attestation timestamp is not greater than the current timestamp on-chain.

At the moment, block metadata cannot be accessed on-chain in Aleo. When sufficient support is available it is recommended to verify this on-chain. In the interim, it is recommended to perform this check in the backend when notarizing a request.

Developers Response The developers have acknowledged the issue but indicated they do not plan to provide a fix due to the current limitations on accessing timestamps in the Aleo blockchain.

5.1.6 V-AOP-VUL-006: Lack of mechanism to revoke invalid or compromised attested data

Severity	Warning	**Commit**	N/A
Type	Maintainability	**Status**	Fixed
Location(s)	`programs/vlink_oracle_v0001.leo`		
Confirmed Fix At	`https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1`, 7fdb1c8		

Description The oracle currently does not provide a mechanism to revoke attested data once it has been submitted on-chain. If data is later found to be invalid, or if the enclave or its signing key is compromised, the data remains accessible to consumers. While an admin pause function can prevent new submissions, it does not address the persistence of historical data that continues to be available under the `{request_hash, attestation_timestamp}` key.

This creates problems for consumers who rely on historical lookups, since they may unknowingly retrieve incorrect or maliciously signed values. Without a way to revoke the invalid attested data, consumers may unknowingly continue to rely on malicious and untrusted reports.

Impact Consumers of the oracle may rely on attested data that is malicious or outdated, either because it was signed with a compromised key or an outdated enclave configuration. The inability to revoke or replace invalid attested data undermines the reliability of the oracle.

Recommendation Implement a mechanism for invalidating attested data once it is known to be untrustworthy. For the latest attestation data, this can be done by explicitly clearing the data, or overwriting the entries associated with a compromised signing key.

For historical requests, maintain an association between the timestamped request hash and the enclave configuration, signer key that delivered the report. This can be done by either including the signer key and the enclave configuration within the timestamped hash, or by maintaining a mapping between the timestamped hash and the `(signer, enclave)` pair that reported the related attested data. This association can then be used to identify and clear out entries associated with a compromised key or unsupported enclave configuration.

Consumers should also be made aware whenever historical data is revoked, so that they can refresh or adjust their queries accordingly.

Developers response The developers added the privileged function `update_historic_data()` which allows modifying either `sgx_attested_data` or `nitro_attested_data`.

5.1.7 V-AOP-VUL-007: Owner can set PCR values to zero enabling acceptance of debug mode nitro reports

Severity	Warning	**Commit**	N/A
Type	Data Validation	**Status**	Acknowledged
Location(s)	`programs/vlink_oracle_v0001.leo:75`		
Confirmed Fix At	N/A		

Description The `set_pcr_values()` function allows the contract owner to set PCR values without validation, including setting them to zero.

PCR values are cryptographic measurements from AWS Nitro enclaves that represent the enclave's identity and runtime integrity. When an enclave is launched in "production mode", PCRs contain non-zero SHA-384 hashes representing the enclave's exact boot image and environment. These values can be verified against known good measurements to ensure enclave integrity.

However, when an enclave is launched in debug mode, Nitro sets all PCR values to 0x00...00 (all zeros). This signals that the enclave is running in a mode where the host access to the enclave memory is not restricted which allows the host to observe and manipulate the enclave execution. Consequently, any attestation document containing zeroed PCRs cannot be considered trustworthy.

Impact The contract validates submitted Nitro reports by comparing extracted PCR values against stored expected values using `assert_eq(pcr_values, report_pcr_values)`. If PCR values are set to zero, an attacker controlling the host could launch the enclave in debug mode, produce a Nitro attestation document with PCRs = 0, and still pass validation.

Recommendation Add validation in transition of finalization of `set_pcr_values()` to reject zero values.

Developer Response The developers have acknowledged the issue but indicated they do not plan to provide a fix:

"We won't be using AWS Nitro. We will only use Intel SGX."

5.1.8 V-AOP-VUL-008: Maintainability concerns

Severity	Info	**Commit**	N/A
Type	Maintainability	**Status**	Fixed
Location(s)	`programs/vlink_oracle_v0001.leo:58`		
Confirmed Fix At	https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1, https://github.com/venture23-aleo/aleo-oracle-contracts/pull/1, d3df0ae, 2e0b44d		

Description The Aleo Oracle program contains some minor maintainability issues that, while not functionally incorrect, reduce clarity and readability. These include:

1. **Incorrect comment:** On line 58, the comment mentions "data attested by sgx" whereas the code pertains to nitro attested data.
2. **Missing comments**: The code handling AWS Nitro attestations lacks explanatory comments on design, assumptions, and invariants, reducing readability. Several validations performed in these transitions rely on implicit knowledge of PCR semantics and layout. This makes it hard for reviewers and developers to understand the intended design and validations.

Impact Poor internal documentation increases the chance of subtle mistakes during refactors, and reduces clarity and readability for reviewers or developers.

Recommendation Fix the inconsistency in the comment mentioned above. And to make the design and validations clear, add precise narrative comments to the Nitro verification functions and transitions.

Developer Response The developers implemented the recommendation.