ETHOS NETWORK - SECURITY REVIEW

Ethos Network.

A security review of Ethos protocol contracts covering rewards, vouching, slashing, reviews, market logic, and supporting access-control utilities.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
Ethos Network	Ethos Protocol	Cosine - Zdravko Hristov - Michael Lett	June 9 - June 25, 2026	July 9, 2026

§ 00

Contents.

74 pages - overview, confidence ranking, scope, 56 findings, disclaimer, and about Guardian.

§ 01

Overview.

Guardian reviewed Ethos Network over a main review and remediation review. The final report includes five Medium, seventeen Low, and thirty-four Informational findings.

AUDIT FIRM	Guardian
CLIENT	Ethos Network
PROJECT	Ethos Protocol
PREPARED BY	Cosine - Zdravko Hristov - Michael Lett
AUDIT WINDOW	June 9 - June 25, 2026
FINAL REPORT	July 9, 2026

REPOSITORY	`trust-ethos/ethos-protocol`
METHODOLOGY	Static - Manual - Test
MAIN REVIEW COMMIT	`2e16c17110929849d15dd1357a68d904c18ada23`
REMEDIATION COMMIT	`2e16c17110929849d15dd1357a68d904c18ada23`

1.1 Confidence Ranking

4/5

High Confidence

Guardian assigns a Confidence Ranking of 4 out of 5. No Critical or High findings were identified, and the review surfaced a manageable set of Medium and Low findings across the core rewards, vouching, slashing, and market flows. The codebase is suitable for deployment after remediations, with periodic review recommended when material changes are made.

1	2	3	4	5
V. LOW	LOW	MED	HIGH	V. HIGH

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	DECLINED	ACK	PARTIAL	RESOLVED
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	**5**	0	0	**2**	0	**3**
Low	**17**	0	0	**9**	0	**8**
Info	**34**	0	0	**26**	0	**8**

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence ASSIGNED	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. Design patterns are sound, and test coverage is strong. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity issues. Varied Low severity findings.
3 Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas such as weak modularity or risky patterns. No critical design flaws were identified, though some patterns could lead to edge-case issues. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2 Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week, or additional unresolved High/Critical findings uncovered in remediation review.
1 Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (>=5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	>=5 High/Critical findings and overall systemic flaws.

§ 03

Scope & Classifications.

The audit covered the Ethos Protocol Solidity code listed in the Notion export, including rewards, market, vouching, review, slashing, pricing, interface, and error modules.

3.1 Severity Matrix

	IMP · HIGH	IMP · MEDIUM	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH	Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.
IMPACT · MEDIUM	Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.
IMPACT · LOW	Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.
LIKELIHOOD · HIGH	Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.
LIKELIHOOD · MEDIUM	Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.
LIKELIHOOD · LOW	Unlikely to ever occur in production.

§ 03

Scope.

Files in scope, grouped by repository. SLOC counts are physical source lines.

trust-ethos/ethos-protocol

28 CONTRACTS · 2,227 SLOC

src/	
AdaptiveLMSRPricing.sol	148
EthosMarket.sol	334
EthosRewards.sol	125
EthosVouchV2.sol	382
EthosWhuffie.sol	77
PositionToken.sol	35
src/utils/	
AccessControlV2.sol	114
MathConstants.sol	5
SlashFreezable.sol	27
src/legacy/	
EthosReview.sol	358
EthosSlash.sol	504
src/interfaces/	
IEthosRewards.sol	3
IEthosVouchV2.sol	3
IEthosVouchV2Slashable.sol	3
IFreezable.sol	4
IReputationPricing.sol	14
ISlashable.sol	4
ITargetStatus.sol	3
src/errors/	
AccessControlV2Errors.sol	4
MarketErrors.sol	16
PositionTokenErrors.sol	4
PricingErrors.sol	7
RewardsErrors.sol	6
SignatureErrors.sol	2
SlashFreezableErrors.sol	3
VouchV2Errors.sol	19
WhuffieErrors.sol	6
src/legacy/errors/	
ReviewErrors.sol	17

§ 04

Findings.

Fifty-six findings across the main and remediation review rounds: five Medium, seventeen Low, and thirty-four Informational. Entries marked In Fix Review in the export are shown as Resolved per final report instruction.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
M-01	Slashing Can Be Avoided	Frontrunning	MEDIUM	ACKNOWLEDGED	p. 13
M-02	Frozen Vouches Can Still Grow	Logical Error	MEDIUM	ACKNOWLEDGED	p. 14
M-03	ceil totalAccruedNotPaid increment to prevent claim()	DoS	MEDIUM	RESOLVED	p. 15
M-04	Composite vouch blocked during lock	Unexpected Behavior	MEDIUM	RESOLVED	p. 16
M-05	Authors can exit stake before slashing	DoS	MEDIUM	RESOLVED	p. 17
L-01	currentRewardRateBps Can Be Stale	Unexpected Behavior	LOW	RESOLVED	p. 18
L-02	Reward Deposits Accrue Retroactively	Logical Error	LOW	RESOLVED	p. 19
L-03	Zero Increase Emits Misleading Event	Events	LOW	RESOLVED	p. 20
L-04	increaseVouch Skips Minimum Check	Validation	LOW	RESOLVED	p. 21
L-05	Paused Markets Allow Token Transfers	Unexpected Behavior	LOW	ACKNOWLEDGED	p. 22
L-06	Attestation Review Index Split	Documentation	LOW	ACKNOWLEDGED	p. 23

§ 04

Findings, continued.

Index page 2 of 6.

§ 04

Findings, continued.

Index page 3 of 6.

§ 04

Findings, continued.

Index page 4 of 6.

§ 04

Findings, continued.

Index page 5 of 6.

§ 04

Findings, continued.

Index page 6 of 6.

MEDIUM M-01 ACKNOWLEDGED

Slashing Can Be Avoided.

CATEGORY Frontrunning LOCATION src/EthosVouchV2.sol#L544-L574 REVIEW Main Review

DESCRIPTION

Users can avoid being slashed if they already know that their behavior has been publicly exposed.

For example:

- Eve vouches on Ethos.
- Eve performs a malicious action for personal gain, hoping that it will never be discovered.
- Alice discovers Eve's activity and publicly reveals it on X.
- Many people, including Bob and Eve, see the statement, and Bob tries to slash Eve on Ethos.
- Eve now still has time until Ethos signs the slash and Bob executes it. In the meantime, she can call `unvouch` to avoid penalties.

RECOMMENDATION

Be aware and consider to make `unvouch` and `decreaseVouch` a 2-Step flow with a time delay to reduce the risk of users exiting the system before they can be slashed.

RESOLUTION

Ethos Network Team - Acknowledged.

MEDIUM M-02 ACKNOWLEDGED

Frozen Vouches Can Still Grow.

CATEGORY Logical Error LOCATION src/EthosVouchV2.sol REVIEW Main Review

DESCRIPTION

Financial slash creation snapshots and freezes the subject/author addresses, but EthosVouchV2 only blocks frozen accounts from `decreaseVouch` and `unvouch`. Frozen accounts can still call `vouch`, `vouchWithPermit`, `vouchFor` via EthosReview, and `increaseVouch`.

At resolution execution, EthosSlash does not burn a snapshotted balance or set of vouch IDs. Instead it calls `EthosVouchV2.slash(addr, bps)` for each snapshotted address, and EthosVouchV2 iterates over that address's current active vouches. As a result, vouches created or increased after the account was frozen but before `executeResolution()` are included in the financial slash.

This will encourage gaming (to vouch with other not connected accounts instead) and decentivize users from vouching.

RECOMMENDATION

Consider to reject `vouch`, `vouchWithPermit`, `vouchFor`, `increaseVouch`, and `increaseVouchWithPermit` when the author is frozen, or to document this behavior.

RESOLUTION

Ethos Network Team - Acknowledged.

MEDIUM M-03 RESOLVED

ceil totalAccruedNotPaid increment to prevent claim().

CATEGORY DoS LOCATION Not specified REVIEW Main Review

DESCRIPTION

`EthosRewards.claim()` can permanently revert for a dominant committed-balance holder, locking their accrued rewards with no recovery path. No theft / over-emission but legitimate rewards become permanently unclaimable.

Per-interval floor rounding in `updateReward` caused accumulated residual such that a dominant holder's `earned()` could exceed `totalAccruedNotPaid` by a few wei (confirmed: 5 wei over 10 daily intervals). The checked subtraction `totalAccruedNotPaid -= reward` then panicked `0x11` and permanently locked that user's rewards - a claim-blocking DoS.

RECOMMENDATION

Switch the global `mulDiv` in `updateReward` from floor to ceil (`Math.Rounding.Ceil`) so `totalAccruedNotPaid ≥ sum of claimable` at every snapshot. Solvency in the other direction (`balance ≥ totalAccruedNotPaid`) is preserved: the increment is capped at `maxIncrement = floor(available * WAD / totalCommitted)` , so `ceil(increment * totalCommitted / WAD) ⩽ available` .

Defense-in-depth: Clamp the claim debit to `Math.min(reward, totalAccruedNotPaid)` so a future rounding regression cannot re-introduce the revert.

RESOLUTION

Ethos Network Team - Resolved.

MEDIUM M-04 RESOLVED

Composite vouch blocked during lock.

CATEGORY Unexpected Behavior LOCATION src/legacy/EthosReview.sol#L349-L350 REVIEW Main Review

DESCRIPTION

`EthosWhuffie._update()` blocks all locked-period transfers unless the transfer is a mint, a burn, or either `from` or `to` is the registered `ETHOS_VOUCH_V2` address. This allows direct `EthosVouchV2.vouch()` flows during the initial non-transferable period, because the user transfers `WHUF` directly to `EthosVouchV2`.

However, `EthosReview.reviewAndVouchWithPermit()` first pulls the combined review fee and vouch gross amount from the user into `EthosReview` with `safeTransferFrom()`. During the lock period, this first transfer is `user → EthosReview`, so neither side is `ETHOS_VOUCH_V2` and `EthosWhuffie._update()` reverts with `TransfersLocked`. As a result, the atomic review-and-vouch EOA flow is unusable until transfers are unlocked, despite the project ADR describing the function as the launch-window path for EOA review + vouch.

RECOMMENDATION

Update the locked-transfer rules so the intended composite path is explicitly supported. For example, `EthosWhuffie._update()` can also exempt the registered `ETHOS_REVIEW` contract when transfers are locked

RESOLUTION

Ethos Network Team - Resolved.

MEDIUM M-05 RESOLVED

Authors can exit stake before slashing.

CATEGORY DoS LOCATION EthosSlash.sol, EthosVouchV2.sol REVIEW Main Review

DESCRIPTION

`FINANCIAL` slashes rely on the slash author having economic skin in the game: `createSlash()` freezes the author's snapshotted profile addresses, and a later `DEFENDED` resolution burns the author's active VouchV2 stake. However, the signed `createSlash()` payload does not bind any minimum author stake or author address snapshot, and `createSlash()` does not verify that the frozen author addresses still hold a minimum slashable balance when the transaction is executed.

This creates a time-of-check/time-of-use gap around Echo signing. An author can obtain a valid `createSlash()` signature while they still have slashable stake, then call `unvouch()` or `decreaseVouch()` before submitting `createSlash()`. Since the account is not frozen until `createSlash()` executes, the withdrawal succeeds and reduces the author's active VouchV2 stake. The author can also remove a stake-holding secondary address from their profile before submitting the signed slash, causing `_recordFinancialSlash()` to snapshot only the remaining current profile addresses.

After this, `createSlash()` still succeeds and freezes the subject, but the author has little or no remaining stake that can be burned if the slash resolves `DEFENDED`. This undermines the documented economic deterrent for frivolous `FINANCIAL` slashes. Because the author no longer has meaningful downside, they do not need to cancel the slash during `cancelWindow`; the subject can remain frozen until the full slash `duration` elapses, and potentially longer if `resolveSlash()` or `executeResolution()` is delayed. The only early release path for the subject is admin cancellation followed by `executeResolution()`.

RECOMMENDATION

Bind and enforce an execution-time author stake requirement for `FINANCIAL` slashes. For example, add a signed `minAuthorCommittedBalance` field to the `createSlash()` payload. During `createSlash()`, after `_recordFinancialSlash()` snapshots the author addresses, sum the committed VouchV2 balance for the exact frozen author addresses and revert if the total is below `minAuthorCommittedBalance`.

`EthosRewards.committedBalance` can be used as the efficient source of active vouch principal if it is kept in sync by `vouch`, `increaseVouch`, `unvouch()`, `decreaseVouch()`, and slash burns. Consider also adding an optional signed `minSubjectCommittedBalance` field so Echo can require the subject-side frozen addresses to still hold enough slashable stake when a financial slash is intended to be economically enforceable. Both minimums can be set to zero for flows where Echo intentionally allows no-stake slashes.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-01 RESOLVED

currentRewardRateBps Can Be Stale.

CATEGORY Unexpected Behavior LOCATION src/EthosRewards.sol#L346-L352 REVIEW Main Review

DESCRIPTION

`currentRewardRateBps()` reports the effective reward rate from `available = rewardToken.balanceOf(address(this)) - totalAccruedNotPaid`. This only subtracts rewards that have already been written into storage by a checkpoint. It does not simulate pending global accrual since `lastUpdateTime`.

RECOMMENDATION

Consider to have `currentRewardRateBps()` account for pending global accrual before computing the available balance.

RESOLUTION

Ethos Network Team - Resolved.

LOW **L-02** RESOLVED

Reward Deposits Accrue Retroactively.

CATEGORY Logical Error LOCATION src/EthosRewards.sol#L323-L324 REVIEW Main Review

DESCRIPTION

EthosRewards is funded by plain ERC20 transfers instead of a deposit function.

Because a direct transfer into the contract does not update `lastUpdateTime`, newly transferred reward tokens are treated as if they were available for the entire elapsed interval since the last checkpoint. For example, if no rewards were available for a week, a treasury/user transfers WHUF into EthosRewards, and then any credit/debit/claim/update path runs, the new WHUF is emitted using a one-week `dt`. If an update had been called immediately before the transfer, the same deposit would only accrue from the post-transfer time. This makes reward distribution dependent on deposit/checkpoint ordering and can over-reward users who were committed before the reward tokens actually arrived.

This can also impact the initial rewards if the `lastUpdateTime` is not refreshed first.

RECOMMENDATION

Consider to add an explicit deposit rewards function that runs `updateReward`, then pulls the tokens via `safeTransferFrom` and emits a `RewardsDeposited` event. Or consider to document this behavior.

If not, initialize `EthosRewards` with `emissionRateBps = 0` and change it to `1980` after the rewards are sent to the contract. This way reward accumulation starts from the `setEmissionRate()` call.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-03 RESOLVED

Zero Increase Emits Misleading Event.

CATEGORY Events LOCATION src/EthosVouchV2.sol#L852-L875 REVIEW Main Review

DESCRIPTION

`EthosVouchV2._increaseVouch()` does not reject `amount == 0`. The caller must be the vouch author, so this is not arbitrary-user event spam, but the author can call `increaseVouch(vouchId, 0)`.

That path computes zero fee and zero gross transfer, calls `_checkTransferIn(0)`, leaves the balance unchanged, and still emits a `VouchIncreased` event.

Frontends, indexers, or analytics that treat every VouchIncreased event as a real top-up can display misleading activity even though no stake was added or break in case the frontend doesn't handle this misleading event badly.

RECOMMENDATION

Consider to reject zero-amount increases or to avoid emitting an event in that case.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-04 RESOLVED

increaseVouch Skips Minimum Check.

CATEGORY Validation LOCATION src/EthosVouchV2.sol#L852-L875 REVIEW Main Review

DESCRIPTION

EthosVouchV2 enforces `configuredMinimumVouchAmount` when creating a new vouch, but `_increaseVouch()` does not enforce that an active vouch's resulting balance is at least the minimum.

This matters after slashing: a slash can reduce an active vouch to a sub-minimum or even zero balance while leaving it unarchived and still present in the author indexes/target uniqueness mapping. The author can then increase the vouch by a tiny amount while keeping the active balance below the configured minimum instead of having to satisfy the same minimum required for fresh vouches.

This breaks the minimum-balance invariant.

RECOMMENDATION

Consider to require the resulting balance to be at least `configuredMinimumVouchAmount` whenever an active vouch is increased. Alternatively, archive and remove vouches that reach (almost) zero during a slash.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-05 ACKNOWLEDGED

Paused Markets Allow Token Transfers.

CATEGORY Unexpected Behavior LOCATION src/EthosMarket.sol#L441-L447 REVIEW Main Review

DESCRIPTION

Pausing a market through EthosMarket only blocks protocol-level open and close operations. The trust and distrust PositionToken contracts remain normal transferable ERC20s, so holders can still transfer or sell those position tokens on third-party markets while the actual Ethos market is paused or frozen.

This means market pause does not fully freeze exposure ownership; it only freezes the protocol's own buy/sell entrypoints.

RECOMMENDATION

Consider to pause this too in that case or to document this behavior.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-06 ACKNOWLEDGED

Attestation Review Index Split.

CATEGORY Documentation LOCATION src/legacy/EthosProfile.sol#L311-L321 REVIEW Main Review

DESCRIPTION

When an unresolved attestation is reviewed, EthosReview stores the review in reviewIdsByAttestationHash[attestationHash]. If that attestation is later linked to an existing profile, EthosProfile.assignExistingProfileToAttestation() only notifies legacy EthosVouch through handleAttestationClaim() before setting profileIdByAttestation. It does not notify EthosReview or migrate/alias the review indexes. As a result, reviews created before the attestation claim remain discoverable through the original attestation-hash index rather than being merged into profile/address-style lookup semantics after the claim.

RECOMMENDATION

Consider to either document that attestation-created reviews remain under the original attestation hash after the attestation is linked to a profile and handle it accordingly in the frontend or add merged lookup/alias behavior.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-07 ACKNOWLEDGED

Identity Remap Enables Self Reviews.

CATEGORY Validation LOCATION src/legacy/EthosProfile.sol REVIEW Main Review

DESCRIPTION

EthosReview prevents self-reviews only against the identity mapping that exists when the review is created. If a user reviews an address that is not registered yet, the review path creates/uses a mock profile for that subject. Later, the author can register that reviewed address as a secondary address on their own real profile. After that registration, current identity resolution maps the reviewed subject back to the author's own profile, so the review violates the intended no-same-profile-review invariant.

RECOMMENDATION

Consider documenting that the no-self-review check can be broken and handle this accordingly in the frontend.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-08 RESOLVED

Wrong error for maximum below minimum.

CATEGORY Error

LOCATION src/EthosVouchV2.sol#L307-L308https://github.com/GuardianOrg/ethos-protocol-team1-1780684149287/blob/774406beeaa04abdac2ab2ea4766668ecc400463/src/EthosVouchV2.sol#L766-L768

REVIEW Main Review

DESCRIPTION

`EthosVouchV2.initialize()` and `EthosVouchV2.setMaximumVouchAmount()` revert with `AmountAboveMaximum` when the provided maximum vouch amount is lower than the configured minimum. This is the inverse of the actual validation failure: the provided value is below the minimum acceptable value. The repository already defines `AmountBelowMinimum(uint256 provided, uint256 minimum)` for this condition, while `AmountAboveMaximum` is documented for vouch creation or increases that would exceed the configured maximum.

RECOMMENDATION

Use `AmountBelowMinimum(configuredMaximumVouchAmount_, configuredMinimumVouchAmount_)` in `initialize()` and `AmountBelowMinimum(amount, configuredMinimumVouchAmount)` in `setMaximumVouchAmount()`.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-09 ACKNOWLEDGED

Duplicate Profile Addresses Persist.

CATEGORY Unexpected Behavior LOCATION src/legacy/EthosProfile.sol#L385-L388 REVIEW Main Review

DESCRIPTION

`EthosProfile.registerAddress()` can append the same address to the same profile multiple times. The function only rejects an address that is already registered to a different profile; if `profileStatusByAddress(addressStr)` resolves to the same `profileId`, it still updates the index mapping and pushes another copy into `profiles[profileId].addresses`.

Because `deleteAddress()` / `deleteAddressAtIndex()` removes only one array entry and then clears `profileIdByAddress[addressStr]`, a duplicate can leave a stale copy in `addressesForProfile(profileId)` even after the address has been unclaimed. Which can lead to weird unexpected behavior in the system.

This requires the trusted signer/backend to issue multiple authorizations for the same address/profile pair, so users cannot create it unilaterally, but the contract does not enforce the uniqueness invariant on-chain.

RECOMMENDATION

Consider to enforce address uniqueness within each profile on-chain. Or document this and be aware to not issue signatures in case the address was already added to the profil.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-10 ACKNOWLEDGED

Asymmetric reviewPrice check.

CATEGORY Validation LOCATION EthosReview.sol REVIEW Main Review

DESCRIPTION

`EthosReview.reviewAndVouchWithPermit()` protects the user against unexpected `reviewPrice` changes, by requiring that `permit.value == actualTotal`.

However, there is no such check in `addReview()` and `addReviewWithPermit()`. If a user has given a large approval to the contract and their review creation transaction lands after an increase of `reviewCost`, they will end up paying more funds than they initially expected.

RECOMMENDATION

Consider adding such check in `addReview()` and `addReviewWithPermit()` as well.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-11 ACKNOWLEDGED

Compromised authors strand review management.

CATEGORY Unexpected Behavior LOCATION src/legacy/EthosReview.sol#L691 REVIEW Main Review

DESCRIPTION

The review management functions execute `_onlyReviewAuthor()`, which validates the `msg.sender`'s current profile against the review `author`'s current profile, as they also have to be verified profiles. This allows the system to keep the intended model where review management authorization follows the current profile of the author. However, if the author address is later deleted from the profile and marked as compromised, `_onlyReviewAuthor()` will keep reverting. As a result, the rest of the addresses in the same profile will not be able to `edit`, `archive` or `restore` the review anymore.

RECOMMENDATION

Consider whether this is intentional behavior. A possible fix may be to add a reassignment function that can be invoked when the author of a review is compromised in order to attach the author to another participant of the same profile.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-12 ACKNOWLEDGED

Cancelled slashes can shield subjects.

CATEGORY DoS LOCATION src/legacy/EthosSlash.sol#L983-L995 REVIEW Main Review

DESCRIPTION

The documentation already acknowledges that a junk slash can shield a subject by occupying the one-open-slash cap slot. This is an accepted tradeoff of the type-agnostic subject cap: `_assertNoOpenSlashes()` rejects a new slash whenever the subject profile, address, or attestation key points to a slash that is still `isOpen()`, regardless of whether the existing slash is `SCORE`, `XP`, or `FINANCIAL`.

The under-documented edge is cancellation. The documented deterrent assumes the shielding slash exposes its author to reputational or financial consequences. However, the shielding author can call `cancelSlash()` during `cancelWindow`. Once `cancelledAt` is set, `isOpen()` returns false, the subject cap slot is freed, and the cancelled slash avoids a terminal negative outcome. The author, or another cooperating author, can then request a new signed slash and occupy the subject slot again.

This creates a cancel-and-replace shielding pattern. A signed low-impact `SCORE` or `XP` slash can temporarily block a later legitimate `FINANCIAL` slash without freezing the subject's VouchV2 stake. A signed `FINANCIAL` shielding slash can also be cancelled into `CANCELLED`, which has no burn, before being replaced. The pattern still requires Echo to sign each slash, and Echo can detect repeated cancellations or same-subject replacement loops, but the on-chain lifecycle lets cancellation avoid the downside that the documented shield-griefing discussion relies on.

RECOMMENDATION

Consider adding a small fee for cancelling a slash so repeated cancel-and-replace shielding has an on-chain cost.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-13 ACKNOWLEDGED

Adaptive LMSR Dust Profit.

CATEGORY Math LOCATION src/AdaptiveLMSRPricing.sol REVIEW Main Review

DESCRIPTION

When an `AdaptiveLMSR` market has zero protocol fees, a user can extract a small amount of `WHUF` by buying the minimum allowed position and then selling nearly the full position while leaving a tiny amount of position-token dust behind.

The issue comes from precision loss in the fixed-point `ln()` and `exp()` operations used by the `AdaptiveLMSR` cost function. `EthosMarket.openPosition()` enforces `MIN_BUY`, but it does not prevent the user from later selling almost all minted position tokens. For the tested production-scale constants, buying `MIN_BUY` and selling all but `256` wei of position tokens returns `MIN_BUY + 1,765` wei, leaving the attacker with both a tiny residual position and a small `WHUF` profit.

The profit is small per execution and is state-dependent. The first sell-down-to-dust moves the market away from its initial supply state, so repeating the exact same sequence against the same market may produce a different residual.

In production this should be offset by the entrance and exit fees.

RECOMMENDATION

Make sure to enable fees for the markets and be aware of this issue.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-14 RESOLVED

Gas bombing via large metadata.

CATEGORY Gas Griefing

LOCATION src/legacy/EthosSlash.sol#L399-L400, src/legacy/EthosSlash.sol#L510-L511, EthosReview.sol

REVIEW Main Review

DESCRIPTION

`editSlash` has no length cap for replacement `comment/metadata`, and status helpers copy the whole `Slash` struct from storage to memory even when they only need scalar fields. An author can therefore store large edited strings and make subsequent status/cancel helper paths for that slash much more expensive.

This will make more expensive not only view functions, but `cancelSlash` as well because it copies the struct as well. The same issue exists in `EthosReview` as well.

RECOMMENDATION

Consider capping slash comment/metadata length on create and edit and doing that in `EthosReview` as well.

RESOLUTION

Ethos Network Team - Resolved.

LOW L-15 ACKNOWLEDGED

Reverting valid buys close to the MAX_SAFE_SUPPLY_SUM.

CATEGORY DoS LOCATION src/AdaptiveLMSRPricing.sol#L262-L275 REVIEW Main Review

DESCRIPTION

`AdaptiveLMSRPricing.getTokensForBudget()` depends on `_findUpperBoundForBudget()` to find an upper token amount for its binary search. In the committed implementation, `_findUpperBoundForBudget()` starts probing at `WAD` and then doubles the probe up to `MAX_TOKENS_PER_TRADE`, without bounding probes by remaining `MAX_SAFE_SUPPLY_SUM` headroom.

When `trustSupply + distrustSupply` is close to `MAX_SAFE_SUPPLY_SUM`, a buy can still be valid through `getCost()`, but `getTokensForBudget()` can revert before considering that amount because the first or a later doubling probe crosses the remaining supply headroom.

This is not limited to sub-`WAD` buys. The general failure condition is that there exists a valid buy amount `x ⩽ headroom`, but `_findUpperBoundForBudget()` probes an amount `y > headroom` before finding a bracket. The sub-`WAD` case is only the smallest repro because the first hardcoded probe is `WAD`; larger headroom values can fail when a later doubling step skips past the remaining headroom.

This affects `EthosMarket.quoteBuy()` and `EthosMarket.openPosition()`, because both use `getTokensForBudget()` to convert a WHUF budget into position tokens.

RECOMMENDATION

A: preserve strict unaffordable upper bound

This is the minimal semantic change. Keep the current postcondition that `_findUpperBoundForBudget()` returns an upper bound whose cost is strictly greater than `budget`:

```
TEXT
1  _buyCostCeil(hi) > budget
```

To do that safely, bound all probes by remaining supply headroom. If the maximum valid probe is still affordable, revert instead of returning `headroom`, because the current `EthosMarket` buy path charges the full budget.

Example patch shape:

LOW L-15 CONTINUED 2/5 ACKNOWLEDGED

Reverting valid buys close to the MAX_SAFE_SUPPLY_SUM.

RECOMMENDATION

SOLIDITY

```solidity
function _findUpperBoundForBudget(
  uint256 trustSupply,
  uint256 distrustSupply,
  bool isPositive,
  uint256 budget
) internal view returns (uint256) {
  if (distrustSupply > type(uint256).max - trustSupply) {
    revert SuppliesExceedArithmeticLimit(trustSupply, distrustSupply, MAX_SAFE_SUPPLY_SUM);
  }

  uint256 supplySum = trustSupply + distrustSupply;
  if (supplySum ⩾ MAX_SAFE_SUPPLY_SUM) {
    revert SuppliesExceedArithmeticLimit(trustSupply, distrustSupply, MAX_SAFE_SUPPLY_SUM);
  }

  uint256 headroom = MAX_SAFE_SUPPLY_SUM - supplySum;
  uint256 maxProbe = Math.min(MAX_TOKENS_PER_TRADE, headroom);

  uint256 hi = Math.min(WAD, maxProbe);
  if (_buyCostCeil(trustSupply, distrustSupply, isPositive, hi) > budget) {
    return hi;
  }

  while (hi < maxProbe) {
    uint256 next = hi * 2;
    if (next > maxProbe) next = maxProbe;

    if (_buyCostCeil(trustSupply, distrustSupply, isPositive, next) > budget) {
      return next;
    }

    hi = next;
  }

  revert BudgetUpperBoundNotFound(budget);
}
```

LOW L-15 CONTINUED 3/5 ACKNOWLEDGED

Reverting valid buys close to the MAX_SAFE_SUPPLY_SUM.

RECOMMENDATION

Tradeoff: if `budget == cost(headroom)`, this conservative version still reverts because there is no strictly unaffordable upper bound inside the safe supply domain. This preserves the existing binary-search postcondition but prevents exact final-headroom buys through the budget inverse.

B: allow exact final headroom

Alternatively, allow `_findUpperBoundForBudget()` to return `maxProbe` when it is exactly affordable:

TEXT

```
_buyCostCeil(hi) > budget
OR
hi == maxProbe && _buyCostCeil(hi) == budget
```

This lets users buy exactly the final safe headroom when their budget exactly matches `cost(headroom)`, while still reverting when `budget > cost(headroom)` to avoid market overcharge.

Example patch shape:

LOW L-15 CONTINUED 4/5 ACKNOWLEDGED

Reverting valid buys close to the MAX_SAFE_SUPPLY_SUM.

RECOMMENDATION

SOLIDITY

```solidity
function _findUpperBoundForBudget(
  uint256 trustSupply,
  uint256 distrustSupply,
  bool isPositive,
  uint256 budget
) internal view returns (uint256) {
  if (distrustSupply > type(uint256).max - trustSupply) {
    revert SuppliesExceedArithmeticLimit(trustSupply, distrustSupply, MAX_SAFE_SUPPLY_SUM);
  }

  uint256 supplySum = trustSupply + distrustSupply;
  if (supplySum ⩾ MAX_SAFE_SUPPLY_SUM) {
    revert SuppliesExceedArithmeticLimit(trustSupply, distrustSupply, MAX_SAFE_SUPPLY_SUM);
  }

  uint256 headroom = MAX_SAFE_SUPPLY_SUM - supplySum;
  uint256 maxProbe = Math.min(MAX_TOKENS_PER_TRADE, headroom);

  uint256 hi = Math.min(WAD, maxProbe);
  uint256 hiCost = _buyCostCeil(trustSupply, distrustSupply, isPositive, hi);
  if (hiCost > budget) {
    return hi;
  }

  if (hi == maxProbe) {
    if (hiCost == budget) return hi;
    revert BudgetUpperBoundNotFound(budget);
  }

  while (hi < maxProbe) {
    uint256 next = hi * 2;
    if (next > maxProbe) next = maxProbe;

    uint256 nextCost = _buyCostCeil(trustSupply, distrustSupply, isPositive, next);
    if (nextCost > budget) {
      return next;
    }

    if (next == maxProbe) {
      if (nextCost == budget) return next;
      revert BudgetUpperBoundNotFound(budget);
    }

    hi = next;
  }

  revert BudgetUpperBoundNotFound(budget);
}
```

LOW L-15 CONTINUED 5/5 ACKNOWLEDGED

Reverting valid buys close to the MAX_SAFE_SUPPLY_SUM.

RECOMMENDATION

If this option is chosen, update the comment in `getTokensForBudget()` :

SOLIDITY

```
// _findUpperBoundForBudget returns either an unaffordable upper bound,
// or the exact maximum safe-domain buy when its cost equals the budget.
```

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-16 RESOLVED

VouchMetadataUpdated not emitted.

CATEGORY Events LOCATION src/EthosVouchV2.sol#L84 REVIEW Main Review

DESCRIPTION

The behavior of the `VouchMetadataUpdated()` event is documented as fired `on a non-empty metadata at creation time and on every successful setVouchMetadata call`.

However, `_vouchAs()` sets the metadata when it creates the vouch without emitting the event.

TEXT

```
1       if (bytes(metadata).length > 0) {
2         vouchMetadata[vouchId] = metadata;
3       }
```

RECOMMENDATION

DIFF

```
1       if (bytes(metadata).length > 0) {
2         vouchMetadata[vouchId] = metadata;
3   +     emit VouchMetadataUpdated(vouchId, metadata);
4       }
```

RESOLUTION

Ethos Network Team - Resolved.

INFO I-01 ACKNOWLEDGED

Misleading _addReview Name.

CATEGORY Informational LOCATION src/legacy/EthosReview.sol#L429-L442 REVIEW Main Review

DESCRIPTION

`EthosReview._addReview` is misleadingly named. The function does not add or persist a review, emit `ReviewCreated`, or increment `reviewCount`; it only resolves an existing subject profile/mock id or creates a mock profile id via `EthosProfile.incrementProfileCount`.

RECOMMENDATION

Consider to rename `_addReview` to a name that describes its actual behavior to follow best practices.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-02 ACKNOWLEDGED

Delegated EOA signer needs ERC-1271 handling.

CATEGORY Informational

LOCATION AUDIT_FAQ.md#L37-L48https://github.com/GuardianOrg/ethos-protocol-team1-1780684149287/blob/774406beeaa04abdac2ab2ea4766668ecc400463/src/legacy/SignatureVerifier.sol#L25-L27, src/utils/SignatureControl.sol#L42-L65

REVIEW Main Review

DESCRIPTION

The new audit FAQ documents that expectedSigner is currently a backend-controlled EOA, Base is the only deployment scope, and there is one authorized CAM-registered Slash contract. Under those assumptions, OpenZeppelin SignatureChecker verifies the signer through the EOA recovery path and raw signatureUsed[signature] tracking is accepted as a documented deployment assumption. This is therefore not an active vulnerability in the current model. It remains a future-model warning: if expectedSigner is later changed to an ERC-1271 contract, delegated EOA with code, multi-chain signer, or non-deterministic signing system, signature validation and replay protection must be revisited.

RECOMMENDATION

Keep this as an informational future-model note. If expectedSigner ever moves away from the current EOA-only model, require ERC-1271 behavior that rejects alternate encodings of the same authorization, bind signatures to the intended domain where appropriate, and track used message digests or structured nonces instead of raw signature bytes.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO **I-03** RESOLVED

Unused Code.

CATEGORY Superfluous Code LOCATION GLOBAL REVIEW Main Review

DESCRIPTION

Unused errors:

- `src/errors/WhuffieErrors.sol ⇒ ZeroAmount(address account);`
- `src/legacy/EthosVouch.sol ⇒ InvalidFeeMultiplier(uint256 newFee);`
- `src/legacy/EthosVouch.sol ⇒ InsufficientProtocolFeeBalance();`
- `src/legacy/errors/ProfileErrors.sol ⇒ error ProfileNotMock(uint256 profileId);`
- `src/legacy/errors/ProfileErrors.sol ⇒ error ZeroAddress();`
- `src/legacy/errors/ReviewErrors.sol ⇒ error MustCreateAttestationFirst();`

Unused State Variables:

- `src/legacy/EthosProfile.sol ⇒ mapping(uint256 ⇒ mapping(uint256 ⇒ uint256)) private acceptedIdIndexByProfileIdAndId;`

Unused Enums:

- `src/legacy/EthosReview.sol ⇒ enum ReviewsBy`

Unused Functions:

- `src/legacy/EthosReview.sol ⇒ _reviewsInRange()`

RECOMMENDATION

Consider to remove unused code to follow best practices.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-04 ACKNOWLEDGED

Legacy increaseVouch Bypasses Pause.

CATEGORY Access Control LOCATION src/legacy/EthosVouch.sol#L509-L541 REVIEW Main Review

DESCRIPTION

The legacy `EthosVouch.increaseVouch()` function does not include the `whenNotPaused` modifier, that is used all over the system.

This weakens the pause mechanism used for emergency situations.

RECOMMENDATION

Consider to add the `whenNotPaused` modifier to follow best practices.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-05 ACKNOWLEDGED

Market Skips Transfer Delta Checks.

CATEGORY Best Practices LOCATION src/EthosMarket.sol#L620 REVIEW Main Review

DESCRIPTION

EthosMarket records market creation backing and buy payments using nominal transfer amounts and does not perform the balance-delta receive check used by EthosVouchV2 and the legacy review fee pull path. The new audit docs explicitly narrow the Market threat model: EthosMarket.token is always the canonical EthosWhuffie proxy, and non-WHUF, fee-on-transfer, rebasing, or modified WHUF-like tokens are out of scope. Under that documented deployment invariant this is not an active accounting vulnerability; it remains a defensive-consistency / future-hardening note because the Market pull paths diverge from comparable receive-delta checks elsewhere in the codebase.

RECOMMENDATION

Consider to add the same balance-delta check pattern used by EthosVouchV2 and other contracts to stick to the own practices.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-06

ACKNOWLEDGED

Renounce can finalize unusable WHUF state.

CATEGORY Best Practices

LOCATION src/EthosWhuffie.sol#L113-L131https://github.com/GuardianOrg/ethos-protocol-team1-1780684149287/blob/774406beeaa04abdac2ab2ea4766668ecc400463/src/EthosWhuffie.sol#L144-L160

REVIEW Main Review

DESCRIPTION

`EthosWhuffie` inherits OpenZeppelin ownership behavior, so the owner can call `renounceOwnership()` at any time. If ownership is renounced while `transfersUnlocked` is still false, no account can later call `unlockTransfers()`, leaving ordinary WHUF transfers permanently locked. Similarly, if ownership is renounced while the token is paused, no account can call `unpause()`, and `_update()` remains blocked by `whenNotPaused`. Because `_authorizeUpgrade()` is also protected by `onlyOwner`, renouncing in either state also removes the ability to upgrade the token implementation to recover.

RECOMMENDATION

Override `renounceOwnership()` in `EthosWhuffie` and allow it only once `transfersUnlocked == true` and `paused() == false`.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-07 ACKNOWLEDGED

Zero-payout sells can quote as executable.

CATEGORY Informational LOCATION src/EthosMarket.sol#L504 REVIEW Main Review

DESCRIPTION

`quoteSell()` can return a positive `curveRevenue` while `netCreditsOut` is zero. This happens when `_computeExitFee()` rounds the fee up to the full `curveRevenue`, for example a 1 wei `curveRevenue` with any nonzero `exitFeeBps`. The executing path is protected because `closePosition()` reverts with `ZeroPayout()` when `netCreditsOut == 0`, but the quote path still returns the positive `curveRevenue` and `exitFee` instead of zeroing the result like it already does for zero amounts or insufficient seller balance. This can mislead frontends, indexers, or off-chain routing code into treating a dust sell as quoted even though it is not executable.

RECOMMENDATION

Update `quoteSell()` to mirror the executable sell conditions. After `_computePayoutBreakdown()` returns, if `netCreditsOut == 0`, return `(0, 0, 0, 0)`.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-08 ACKNOWLEDGED

Inaccurate _requireActiveMarket() NatSpec.

CATEGORY Documentation LOCATION src/EthosMarket.sol#L719 REVIEW Main Review

DESCRIPTION

The NatSpec of `EthosMarket._requireActiveMarket()` says it's called by `openPosition()`, but the actual function that calls it is `_openPosition()` (with underscore), while `openPosition()` is a wrapper around it.

RECOMMENDATION

DIFF

```
1  - /// @dev Reverts if the market is paused or sell-only. Used by openPosition.
2  + /// @dev Reverts if the market is paused or sell-only. Used by _openPosition.
```

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-09 ACKNOWLEDGED

MIN_BUY purpose is misstated.

CATEGORY Documentation LOCATION src/EthosMarket.sol#L68 REVIEW Main Review

DESCRIPTION

The comment above the `MIN_BUY` constant in `EthosMarket` says it's purpose is to prevent attacks where fees round to 0.

TEXT

```
    /// @notice Minimum buy amount to prevent dust attacks where fees round to zero.
    uint256 public constant MIN_BUY = 1e15; // 0.001 WHUF
```

Even without `MIN_BUY` the fees won't round to 0 because both functions use `Ceil` when calculating them.

TEXT

```
    function _computeEntryFee(uint256 paymentAmount) internal view returns (uint256) {
      return Math.mulDiv(paymentAmount, entryFeeBps, BPS_DENOMINATOR, Math.Rounding.Ceil);
    }

    /// @dev Computes the exit protocol fee from curve revenue.
    function _computeExitFee(uint256 curveRevenue) internal view returns (uint256) {
      return Math.mulDiv(curveRevenue, exitFeeBps, BPS_DENOMINATOR, Math.Rounding.Ceil);
    }
```

This means the comment is not correct about the goal of the constant.

RECOMMENDATION

Consider changing the comment.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-10 ACKNOWLEDGED

Buy trades overchage.

CATEGORY Informational LOCATION src/EthosMarket.sol#L519-L526 REVIEW Main Review

DESCRIPTION

`_openPosition()` computes `tokensMinted` from the full post-fee `curveAmount` budget and transfers the entire `curveAmount` even when the exact cost of `tokensMinted` is lower than the budget. This means unused buy budget is intentionally retained as market backing instead of refunded. The behavior is conservative for solvency and can provide a buffer against rounding and fixed-point approximation effects in the pricing contract, but it is not clearly documented for users or integrators. As a result, callers may incorrectly assume `quoteBuy()` or `openPosition()` charge only the exact bonding-curve cost of the minted tokens.

RECOMMENDATION

Consider documenting this overcharging behavior.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-11 ACKNOWLEDGED

Frozen event is emitted without state change.

CATEGORY Informational LOCATION src/utils/SlashFreezable.sol#L59-L67 REVIEW Main Review

DESCRIPTION

`IFreezable.Frozen` is documented as emitted when an account's frozen state changes, but `SlashFreezable.freeze()` and `SlashFreezable.unfreeze()` emit the event every time the status is set. Repeated calls such as `freeze(account)` when the account is already frozen, or `unfreeze(account)` when the account is already unfrozen, emit `Frozen` without an actual state transition.

RECOMMENDATION

Either update the NatSpec to describe `Frozen` as a status-set event rather than a state-change event, or make `freeze()` and `unfreeze()` emit only when `_frozenAccounts[account]` actually changes.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-12 ACKNOWLEDGED

Missing gap in signature control.

CATEGORY Upgradeability LOCATION src/utils/SignatureControl.sol#L13-L17 REVIEW Main Review

DESCRIPTION

`SignatureControl` is an upgradeable storage-bearing base contract, but it does not reserve a storage gap after its state variables. It currently stores `expectedSigner`, `signatureVerifier`, and `signatureUsed`, and child contracts such as `AccessControlV2` begin their own storage immediately after those slots. As a result, adding new storage variables to `SignatureControl` in a future upgrade would shift child storage and collide with existing variables such as `contractAddressManager` and `pendingOwner`. This limits safe upgradeability of contracts inheriting `SignatureControl`. However, the `SignatureControl` is currently in use by the deployed legacy contracts, so adding a gap to it would shift the current storage layout.

RECOMMENDATION

Acknowledge the finding and be aware of it in case you need to add a new variable to `SignatureControl` in a future upgrade.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-13 ACKNOWLEDGED

Archived Targets Stay Allowed.

CATEGORY Unexpected Behavior LOCATION src/legacy/EthosReview.sol#L535-L540 REVIEW Main Review

DESCRIPTION

`ITargetStatus.targetExistsAndAllowedForId()` is used by many contracts to implement a function that checks if a given object exists and it is allowed to be used.

`EthosSlash` uses both parts of the status: an existing slash is only `allowed` while it is still open (`allowed = exists && isOpen(_targetId)`).

By contrast, `EthosReview` returns `allowed = exists` and does not account for `review.archived` . The vouch implementations follow the same existence-implies-allowed pattern for archived vouches, but `EthosVouchV2` documents that archived vouches remain valid discussion targets for continuity. `EthosReview` has no equivalent documentation.

RECOMMENDATION

Clarify the intended lifecycle semantics for archived reviews. If archived reviews should remain valid discussion targets, document that, otherwise update the `targetExistsAndAllowedForId` flow accordingly.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-14 ACKNOWLEDGED

Inconsistent review existence checks.

CATEGORY Error LOCATION src/legacy/EthosReview.sol#L459-L463 REVIEW Main Review

DESCRIPTION

`EthosReview.editReview()` does not perform an existence check before calling `_onlyReviewAuthor()`. For a nonexistent review, `reviews[reviewId].author` is `address(0)`, so `_onlyReviewAuthor()` calls `verifiedProfileIdForAddress(address(0))` and reverts from `EthosProfile` with `ProfileNotFoundForAddress(address(0))` instead of the expected `ReviewNotFound()` error used by the other review-management functions.

This does not allow nonexistent reviews to be edited, but it creates inconsistent error semantics across related review-management functions and makes failed calls harder to diagnose.

RECOMMENDATION

Add an explicit existence check to `editReview()` before `_onlyReviewAuthor()`, using the same `ReviewNotFound()` error as the other review-management functions.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-15 RESOLVED

Comment misstates permit allowance.

CATEGORY Documentation LOCATION src/legacy/EthosReview.sol#L360-L363 REVIEW Main Review

DESCRIPTION

The comment before the `EthosVouchV2` approval in `EthosReview.reviewAndVouchWithPermit()` states that the `permit` value lock guarantees no residual allowance. This is misleading because the consumed `permit` applies to the user-to-`EthosReview` allowance, while the later `forceApprove()` creates a separate `EthosReview`-to-`EthosVouchV2` allowance. The absence of residual `EthosVouchV2` allowance in the current implementation comes from approving exactly `vouchGross`, `EthosVouchV2._checkTransferIn()` pulling exactly that amount, and the explicit `forceApprove(..., 0)` cleanup, not from the user's `permit` value lock.

RECOMMENDATION

Update the comment to describe the actual allowance relationship. For example: "Approve `EthosVouchV2` for exactly `vouchGross`, then call `vouchFor()`. The current `EthosVouchV2._checkTransferIn()` pulls the full approved amount, and the allowance is reset to zero defensively in case a future `EthosVouchV2` implementation pulls less than approved."

RESOLUTION

Ethos Network Team - Resolved.

INFO I-16 ACKNOWLEDGED

Inconsistent delta check.

CATEGORY Validation LOCATION src/legacy/EthosReview.sol#L350 REVIEW Main Review

DESCRIPTION

When a review is created, `_pullAndBurnReviewFee()` reverts if the used token has fee on transfers enabled by comparing the transferred delta against the expected amount.

However, `reviewAndVouchWithPermit()` pulls the user funds without calling `_pullAndBurnReviewFee()`, effectively skipping the delta check.

RECOMMENDATION

Consider applying the same delta check in `reviewAndVouchWithPermit()`

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-17 RESOLVED

Missing authorProfileId in Review struct.

CATEGORY Documentation LOCATION src/legacy/EthosReview.sol#L98 REVIEW Main Review

DESCRIPTION

The NatSpec of the `Review` struct in `EthosReview.sol` documents an `authorProfileId` field which is not present in the struct.

RECOMMENDATION

Remove the field from the NatSpec.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-18 ACKNOWLEDGED

vouchIdsByAuthorIndex has ambiguous name.

CATEGORY Best Practices LOCATION src/EthosVouchV2.sol#L245 REVIEW Main Review

DESCRIPTION

The `vouchIdsByAuthorIndex` variable holds the index of each `vouchId` in the `vouchIdsByAuthor[address]` array, but its name suggests that it is holding vouch IDs.

RECOMMENDATION

Consider renaming the variable. For example, `indexesByUserVouchIds`.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-19

ACKNOWLEDGED

Zero increments can suppress rewards.

CATEGORY Rounding LOCATION src/EthosRewards.sol#L316-L324 REVIEW Main Review

DESCRIPTION

When `totalCommitted` is very large relative to the remaining reward pool, `rewardPerToken()` can compute an `increment` of `0` after integer division. The `updateReward()` modifier still updates `lastUpdateTime` when `newRpt == oldRpt`, so each zero-increment checkpoint permanently discards the elapsed time instead of allowing it to accumulate into a later nonzero increment.

A user with an existing vouch can trigger these checkpoints through tiny nonzero `increaseVouch()` calls, which call `credit()` on `EthosRewards`. Under sufficiently high `totalCommitted / available` ratios, repeated small checkpoints can suppress rewards for all users. The PoC test `test_audit_dailyTinyCreditsCanSuppressYearlyRewardsWhenCommittedDwarfsPool()` shows daily 1-wei `credit()` calls for one year leaving `rewardPerTokenStored`, `totalAccruedNotPaid`, and `earned()` at `0`, while the same state with a sparse one-year checkpoint accrues approximately the expected annual rewards.

The required ratio is extreme. At the documented `1980` bps emission rate, daily checkpoints require `totalCommitted / available` above roughly `5.42e14`. This makes the issue highly dependent on the production WHUF supply, reward pool size, and expected committed balance.

RECOMMENDATION

A possible mitigation is to avoid advancing `lastUpdateTime` only when `totalCommitted > 0`, `dt > 0`, `emissionRateBps > 0`, `available > 0`, and `increment == 0`.

This mitigation has a tradeoff: the next nonzero checkpoint may allocate the carried interval across the then-current committed set, so users who joined during the carried interval can receive a small portion of delayed rewards. The carried interval is bounded by the time needed for `increment` to become nonzero.

Either acknowledge the current behavior or implement the suggested fix, depending on which behavior is acceptable.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-20 ACKNOWLEDGED

Slash accepts partial attestations.

CATEGORY Validation LOCATION src/legacy/EthosSlash.sol#L874-L879 REVIEW Main Review

DESCRIPTION

`EthosSlash._validateSubjectOrAttestationSet()` treats an attestation as present when either `attestationDetails.account` or `attestationDetails.service` is non-empty. This is more permissive than `EthosReview._validateReviewDetails()`, which requires both fields when the subject address is unset. The malformed slash is later rejected indirectly because `EthosAttestation.getServiceAndAccountHash()` reverts when either field is empty, so a slash is not recorded with a partial attestation. However, the slash contract's own validation accepts an invalid attestation shape and relies on a downstream external contract for the final rejection, producing inconsistent validation behavior and revert sources across review and slash flows.

RECOMMENDATION

Validate attestation details directly in `EthosSlash._validateSubjectOrAttestationSet()` using the same rule as `EthosReview`: when `subject` is `address(0)`, require both `attestationDetails.account` and `attestationDetails.service` to be non-empty; when `subject` is set, require both fields to be empty. This keeps malformed slash inputs rejected at the slash validation layer with `InvalidSlashDetails`.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-21 ACKNOWLEDGED

VouchRequiresPositiveReview semantics.

CATEGORY Validation LOCATION src/legacy/EthosReview.sol#L325 REVIEW Main Review

DESCRIPTION

The `VouchRequiresPositiveReview` is thrown when a `reviewAndVouchWithPermit()` call tries to create a non-positive review while vouching. However, since `The composite intentionally does not enforce` vouchUserkey `↔` (subject, attestationDetails)`, the review target may be different than the vouch recipient.

The check successfully guards against non-positive reviews, even though these reviews may not have been given to the same vouch target.

RECOMMENDATION

Consider whether this positive check is needed.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-22 ACKNOWLEDGED

Lifecycle events omit attestation target.

CATEGORY Events LOCATION src/legacy/EthosReview.sol#L454-L508 REVIEW Main Review

DESCRIPTION

`ReviewCreated` includes both target forms by emitting `attestationHash` and `subject`, but `ReviewEdited`, `ReviewArchived`, and `ReviewRestored` only emit `subject`. When a review targets an attestation, `review.subject` is `address(0)`, so lifecycle events emitted by `editReview()`, `archiveReview()`, and `restoreReview()` do not identify the reviewed attestation. Event-only consumers must reconstruct the target from prior `ReviewCreated` logs or call `reviews()` and recompute the hash from `attestationDetails`, which can make attestation review updates unattributable if lifecycle events are processed independently.

RECOMMENDATION

Consider including the attestation target in review lifecycle logs

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-23 ACKNOWLEDGED

Edit slash uses generic expiry error.

CATEGORY Error LOCATION src/legacy/EthosSlash.sol#L503-L508 REVIEW Main Review

DESCRIPTION

`editSlash()` reverts with `EditWindowExpired` whenever `isEditable()` returns false. However, `isEditable()` returns false for multiple distinct states, including cancelled slashes, slashes closed by duration, and slashes whose edit window has elapsed.

This differs from `cancelSlash()`, which disambiguates the same categories and can return `SlashIsCancelled`, `SlashIsClosed`, or `CancelWindowExpired`. As a result, callers and integrators receive a generic edit-window error even when the slash is no longer editable because it was cancelled or closed, not because the edit window alone expired.

RECOMMENDATION

Mirror the explicit branching used by `cancelSlash()` so `editSlash()` returns state-specific errors such as `SlashIsCancelled`, `SlashIsClosed`, and `EditWindowExpired`.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-24 ACKNOWLEDGED

Author stake can back multiple slashes.

CATEGORY Trust Assumptions LOCATION src/legacy/EthosSlash.sol#L983-L985 REVIEW Main Review

DESCRIPTION

`EthosSlash` enforces the author-side concurrent slash cap by the caller's current `authorProfileId`. `_assertNoOpenSlashes()` checks `lastSlashIdByAuthorProfile[authorProfileId]`, and `createSlash()` later snapshots the current addresses of that profile for `FINANCIAL` author-side freezing.

This profile-level cap does not explicitly account for address-level stake already encumbered by another financial slash. If an address that was included in one financial slash's author snapshot is later moved to another profile, that address can be included in another profile's author snapshot for a different slash. `_freezeRefCount` keeps freeze/unfreeze accounting correct, but the same address-level VouchV2 stake can still serve as author-side economic backing for multiple concurrent slashes.

For example, profile `P1` contains addresses `A` and `B`, and `A` creates a financial slash. The author snapshot includes both `A` and `B`, so both addresses are frozen as author-side backing. If `B` is later removed from `P1` and added to profile `P2`, a new slash created from `P2` can snapshot `B` again as author-side backing for a different subject, even though `B` is already frozen by the first slash.

RECOMMENDATION

Consider this case and add the necessary guards in the offchain signing flow if not already present.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-25 RESOLVED

Global slash windows affect open slashes.

CATEGORY Configuration LOCATION src/legacy/EthosSlash.sol#L323-L346 REVIEW Main Review

DESCRIPTION

`EthosSlash` snapshots `duration` in each `Slash`, but `isEditable()` and `isCancellable()` use the current global `editWindow` and `cancelWindow` instead of per-slash values. As a result, admin calls to `setEditWindow()` or `setCancelWindow()` retroactively affect every still-open slash: increasing a window can make an older slash editable or cancellable again after its previous window elapsed, while decreasing a window can remove author rights earlier than expected.

The setters only validate the new windows against the current `defaultDuration`, not against already-created slashes' stored `duration`. `isClosed()` still bounds both operations by each slash's stored `duration`, so this cannot extend editing or author cancellation beyond slash expiry. The issue is that slash window semantics are live global policy rather than per-slash snapshots, which may surprise integrators and users expecting timing terms to be fixed when the slash is created.

RECOMMENDATION

If slash timing terms are expected to be immutable after creation, snapshot `editWindow` and `cancelWindow` into each `Slash` during `createSlash()` and have `isEditable()` and `isCancellable()` read the per-slash values. If the live-global behavior is intentional, document that `setEditWindow()` and `setCancelWindow()` retroactively affect all still-open slashes.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-26 ACKNOWLEDGED

Slash cap can be violated during migration.

CATEGORY Upgradeability LOCATION EthosSlash.sol REVIEW Main Review

DESCRIPTION

The newly added slash cap may be temporarily violated during the upgrade of the previous `EthosSlash` to the new one. This can happen because `XP/SCORE` slashes created before the upgrade won't be populated in the slash cap mappings and `_assertNoOpenSlashes()` will allow opening a new slash even if a previous `XP/SCORE` one already exists. The issue exists for `FINANCIAL` type as well, but these were never supported before the upgrade.

This shouldn't be big of a problem, given that the codebase was working without caps until now. After the old slashes resolve or are cancelled, the system caps should continue working as intended.

RECOMMENDATION

Keep this behavior in mind while upgrading the system.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-27 RESOLVED

Misleading slash signature comment.

CATEGORY Documentation

LOCATION Ethos%20Network%20Main%20Review%20Findings%20Table%20%5BSeverity/I-27%203868bda5828c81f28938c091de4c069c.html

REVIEW Main Review

DESCRIPTION

The comment before `validateAndSaveSignature()` says that signature verification is performed before the slash cap check so unauthenticated callers cannot probe open-slash state by observing `SubjectHasOpenSlash` versus `InvalidSignature`. This privacy rationale is misleading because the cap state is already publicly observable through the public `lastSlashIdByAuthorProfile`, `lastSlashIdBySubjectProfile`, `lastSlashIdBySubjectAddress`, and `lastSlashIdBySubjectAttestationHash` mappings combined with `isOpen()`.

The comment also says that a cap-conflicted submission burns the signature. However, if `_assertNoOpenSlashes()` reverts after `validateAndSaveSignature()` marks `signatureUsed[signature]`, the entire transaction reverts, including the signature usage write. Therefore the signature is not consumed by a cap-conflict revert and remains reusable until it expires or is successfully consumed.

RECOMMENDATION

Update or remove the comment so it accurately reflects the behavior of `createSlash()`.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-28 ACKNOWLEDGED

Uninitialized proxies can be hijacked.

CATEGORY Upgradeability LOCATION Global REVIEW Main Review

DESCRIPTION

If the deployment of the V2 contracts and the new versions of the legacy contracts is not atomic, a malicious actor can call their `initialize()` functions and initialize them with values controlled by them. If unnoticed, this can lead to stolen funds, especially if the malicious user has ownership rights.

RECOMMENDATION

Execute the deployment + initialization in one tx.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-29 ACKNOWLEDGED

Address deletion can shed review history from live profile membership.

CATEGORY Unexpected Behavior LOCATION EthosReview.sol REVIEW Main Review

DESCRIPTION

`EthosReview` resolves or creates a subject profile/mock id when writing a review, but the Review row and subject lookup are stored only by the mutable subject address. EthosProfile can later remove that address from the profile's live address set by clearing profileIdByAddress, without updating any review index or retaining a profile-level review binding.

A profile can delete a negatively reviewed secondary address and continue with its remaining addresses, while consumers using the protocol's live addressesForProfile plus reviewIdsBySubjectAddress surfaces no longer see that review as part of the profile

RECOMMENDATION

**Keep than in mind when handling reviews offchain.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-30 RESOLVED

One-way WHUF transfer unlock emits no event.

CATEGORY Events LOCATION EthosWhuffie.sol REVIEW Main Review

DESCRIPTION

EthosWhuffie.unlockTransfers() permanently flips transfersUnlocked from false to true but emits no event for that irreversible configuration change. The only on-chain signal is the storage getter/state diff rather than a standard log subscription.

IMPACT

Event-driven indexers, monitoring, frontends, and other off-chain integrations can miss or lag the WHUF transferability transition unless they poll storage or inspect every transaction trace.

RECOMMENDATION

Emit a dedicated event such as TransfersUnlocked(address indexed owner) immediately after setting transfersUnlocked = true.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-31 ACKNOWLEDGED

Market buys lack an explicit trade deadline.

CATEGORY Validation LOCATION EthosMarket.sol REVIEW Main Review

DESCRIPTION

EthosMarket buy paths only bound execution with paymentAmount and minTokensOut. They do not accept a separate orderDeadline, and the deadline in openPositionWithPermit is only the ERC-20 permit expiry. A still-valid user transaction can therefore execute later than the user's intended trading window if the output amount remains above minTokensOut.

IMPACT

This is a Low-severity freshness/UX risk: it does not let third parties spend a victim's permit or bypass msg.sender ownership, but a delayed user transaction can buy at a materially different market state than the wallet quote or off-chain intent assumed.

RECOMMENDATION

Add an explicit orderDeadline parameter to buy paths and revert when block.timestamp exceeds it, or document clearly that minTokensOut is only an amount-slippage bound and does not expire market orders.

RESOLUTION

Ethos Network Team - Acknowledged.

LOW L-01 RESOLVED

Reward Rate Rounding Mismatch.

CATEGORY Rounding LOCATION src/EthosRewards.sol#L383 REVIEW Remediation Review

DESCRIPTION

`EthosRewards.updateReward()` increments `totalAccruedNotPaid` using `Math.Rounding.Ceil` now. However, `currentRewardRateBps()` still simulates pending `accruedNotPaid` with the default floor rounding.

RECOMMENDATION

Use the same `Math.Rounding.Ceil` mode in `currentRewardRateBps()` when simulating pending `accruedNotPaid`, so the view function matches the actual accounting path.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-01 ACKNOWLEDGED

L-14 Not Fully Fixed.

CATEGORY Validation LOCATION src/legacy/EthosSlash.sol#L520-L521 REVIEW Remediation Review

DESCRIPTION

The status/cancel helper gas issue appears mitigated because slash helpers now avoid copying the full Slash struct, but the contracts still do not cap comment / metadata sizes.

RECOMMENDATION

Consider to add explicit maximum length checks for slash/review comment and metadata on both create and edit paths.

RESOLUTION

Ethos Network Team - Acknowledged.

INFO I-02 RESOLVED

Vouch ID gap due to initial offset.

CATEGORY Error LOCATION EthosVouchV2.sol REVIEW Remediation Review

DESCRIPTION

`EthosVouchV2.initialize()` allows `vouchCount` to start from `initialVouchCount_`, which is used as an ID offset. When this value is nonzero, IDs below or equal to the initial offset are holes: they satisfy `vouchId ⩽ vouchCount`, but no `Vouch` was ever written for them.

Several mutating paths use only `vouchId == 0 || vouchId > vouchCount` as the existence check before reading `vouches[vouchId]`. For hole IDs, the default `Vouch` has `author == address(0)`, so calls such as `decreaseVouch()`, `setVouchMetadata()`, `_unvouch()`, and `_increaseVouch()` pass the `VouchNotFound` check and then revert with `UnauthorizedVouchAccess` instead. This weakens error semantics and can confuse callers/indexers, but no direct state corruption or authorization bypass is apparent because `msg.sender` cannot equal `address(0)`.

RECOMMENDATION

Either acknowledge or add an existence helper that also checks `vouches[vouchId].author ≠ address(0)` and use it in the affected mutating paths.

RESOLUTION

Ethos Network Team - Resolved.

INFO I-03 RESOLVED

Locked whuffie can be used to pay review fees.

CATEGORY Best Practices LOCATION EthosWhuffie.sol REVIEW Remediation Review

DESCRIPTION

`EthosWhuffie` aims to whitelist transfers for the composite review-vouch composite flow, so its `_update()` function was updated as it follows:

TEXT

```
        bool isReviewCompositePull = to == review && _msgSender() == review;
        if (!isVouchTransfer && !isReviewCompositePull) revert TransfersLocked(from, to);
```

The flag is named `isReviewCompositePull` and the contract NatSpec says `EthosReview.reviewAndVouchWithPermit are permitted through ContractAddressManager-resolved exemptions`. However, the check cannot differentiate between the composite flow and any other transfer to the `Review` contract where it's the `msg.sender` as well. As a result, `WHUFFIE` can be used to pay for `addReview` and `addReviewWithPermit` fees when `transfersUnlocked` is still `false`. Previously, these calls would have reverted, which required 0 fees in order to create reviews in this lock period.

RECOMMENDATION

If the current code behavior is correct, clarify it in the comments and consider renaming the `bool` flag.

RESOLUTION

Ethos Network Team - Resolved.

§ 05

Disclaimer.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian's position is that each company and individual are responsible for their own due diligence and continuous security. Guardian's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis.

Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract's safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian.

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

WEB

guardianaudits.com

AUDITS

github.com/GuardianAudits/Audits

CONTACT

audits@guardianaudits.com