Audit Report

2026-01-11

Produced for
Immuni Software Pte Ltd

1. Scope

This report presents a security review of the Solidity smart contract code corresponding to (i) the proxy contract deployed at address `0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5` on Ethereum mainnet and (ii) its implementation contract deployed at address `0x39EaCeF0Ee562109dF742C989E9D5227d4C17d2e` on Ethereum mainnet, including any proxy/implementation-linked contracts and dependencies that are part of the verified source packages for those addresses at the time of review.

For clarity, this review does not include a security assessment of the separately deployed `ProxyAdmin` instance configured as the proxy's admin (beyond identifying its address/current owner and observing basic ownership and upgrade-related event history at the reference block).

On-chain configuration and read-only calls in this report are anchored to Ethereum mainnet block 24210396 (2026-01-11 08:53:35 UTC). Historical on-chain events are referenced by their emitting transaction hash and block number.

2. Out of Scope

- Operational security of the privileged contract account (e.g., multisig) and its signers (key management, signing processes, etc.)
- Governance processes and policies for upgrades, minting, and pausing
- The separately deployed proxy admin contract instance and its ownership/security (beyond identifying its current owner and observing basic ownership-transfer and upgrade-related event history)
- Token distribution, market operations, and off-chain systems
- Security of third-party infrastructure (RPC providers, Etherscan, indexers)

3. Executive Summary

The Immunefi token is deployed using the transparent proxy pattern. A `TransparentUpgradeableProxy` delegates calls to an `IMU` implementation contract.

The `IMU` implementation is built on a small set of widely used OpenZeppelin modules:

- `ERC20Upgradeable`
- `AccessControlUpgradeable`
- `ERC20PausableUpgradeable`
- `ERC20BurnableUpgradeable`
- `ERC20PermitUpgradeable`

The custom contract logic surface area is small (`initialize`, `mint`, `pause`, `unpause`, and an `_update` override).

The review was performed against the verified source packages for the proxy and implementation addresses at the time of review. We also validated that the analyzed sources correspond to the deployed on-chain bytecode for both deployments.

In addition to manual review of the small custom logic surface area, the review includes bytecode reproduction, third-party dependency provenance checks, mainnet-fork functional tests (ERC-20 + extensions + proxy behavior), and focused on-chain history review (upgrades, role changes, pause usage, and supply issuance).

No Critical, High, or Medium severity issues were identified. Findings are limited to role-based privileged actions (minting and pausing), pause semantics (pausing blocks transfers, minting, and burning), upgradeability trust assumptions, and minor integration/operational nuances (approvals and permits remain callable while paused).

Overall risk: Low (code-level), assuming privileged role holders are operated securely and as intended.

4. System Overview (as implemented)

4.1 Proxy architecture

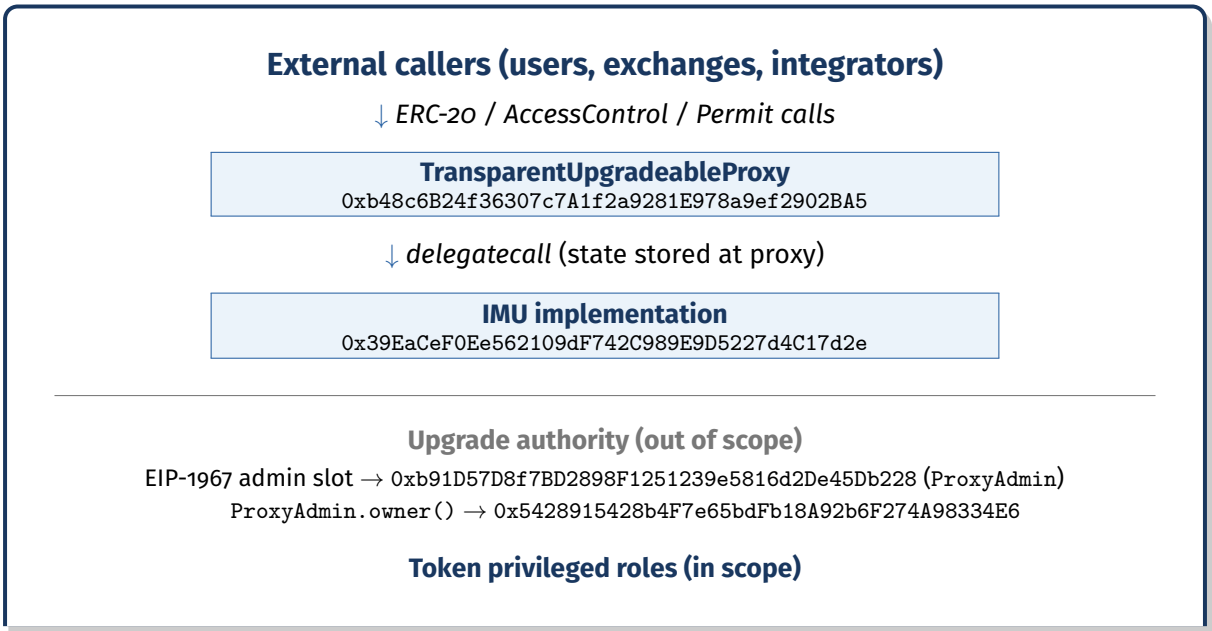
The proxy uses OpenZeppelin's transparent proxy pattern (`TransparentUpgradeableProxy`) with an associated `ProxyAdmin`.

This architecture is upgradeable by design. Governance and operational security of the upgrade authority is out of scope (see Section 2). The following relationships reflect Ethereum mainnet at the audit reference block 24210396 (2026-01-11 08:53:35 UTC):

The transparent proxy pattern enforces that calls from the admin address are not delegated to the implementation. In this deployment the admin is the dedicated `ProxyAdmin` contract, which is only used to perform upgrades via `upgradeAndCall`; all regular token interactions are performed by calling the proxy from non-admin accounts.

Component	Description
Proxy (entrypoint)	`0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5` (`TransparentUpgradeableProxy`)
Implementation (EIP-1967 slot)	`0x39EaCeF0Ee562109dF742C989E9D5227d4C17d2e` (`IMU`)
Proxy admin (EIP-1967 slot)	`0xb91D57D8f7BD2898F1251239e5816d2De45Db228` (`ProxyAdmin`; out of scope)
ProxyAdmin owner()	`0x5428915428b4F7e65bdFb18A92b6F274A98334E6` (out of scope)
Token roles (AccessControl)	`DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, `MINTER_ROLE` held by `0x5428915428b4F7e65bdFb18A92b6F274A98334E6`
State storage	ERC-20 state (balances, allowances, nonces, roles, pause state) is stored at the proxy address; the implementation contract provides logic via `delegatecall`

System diagram (simplified)



```
DEFAULT_ADMIN_ROLE, PAUSER_ROLE, MINTER_ROLE
→ 0x5428915428b4F7e65bdFb18A92b6F274A98334E6
```

The proxy emitted `AdminChanged(address,address)` in the deployment transaction `0x60d9d2e2f4a730dd830430d672da177069b37d28b9eec398365cc784087952ac` at block `23224867`, setting the admin from `address(0)` to `0xb91D57D8f7BD2898F1251239e5816d2De45Db228`. At the audit reference block 24210396, the EIP-1967 admin slot contains `0xb91D57D8f7BD2898F1251239e5816d2De45Db228`. The `ProxyAdmin` contract instance itself (including ownership and any upgrade activity) is out of scope (see Section 2).

The proxy also emitted `Upgraded(address)` in the same deployment transaction, setting the initial implementation to `0x39EaCeF0Ee562109dF742C989E9D5227d4C17d2e`.

No additional proxy `AdminChanged` or `Upgraded` events were observed through block 24210396.

4.2 Token behavior

The `IMU` implementation contract provides an upgradeable ERC-20 with:

- `DEFAULT_ADMIN_ROLE`: admin of all roles (grant/revoke)
- `MINTER_ROLE`: `mint(to, amount)` mints arbitrary amounts
- `PAUSER_ROLE`: `pause()/unpause()` gates transfers, minting, and burning
- `ERC20BurnableUpgradeable`: holders can burn their own tokens and burn from allowances
- `ERC20PermitUpgradeable`: EIP-2612 permits for approvals by signature

The token does not implement custom transfer mechanics such as transfer fees, blacklists, whitelists, or anti-bot restrictions. Aside from the pause gate, it inherits OpenZeppelin's standard ERC-20 behavior.

Initialization safety The `IMU` contract follows OpenZeppelin's upgradeable initialization pattern via `Initializable`. The implementation contract's constructor calls `_disableInitializers()`, which permanently locks the implementation instance against being initialized directly. This is an OpenZeppelin-recommended best practice to prevent accidental initialization of the implementation contract (sometimes referred to as an "implementation takeover") and to reduce the chance that users or integrators accidentally interact with the implementation address instead of the proxy.

During deployment (transaction `0x60d9d2e2f4a730dd830430d672da177069b37d28b9eec398365cc784087952ac` at block `23224867`), the token emitted `RoleGranted` events granting all three roles to `0x5428915428b4F7e65bdFb18A92b6F274A98334E6`.

At the audit reference block 24210396, this address holds these roles.

4.3 On-chain configuration snapshot

The following values were read via the proxy at `0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5` on Ethereum mainnet at block 24210396 (2026-01-11 08:53:35 UTC). This is a point-in-time snapshot and may change if the implementation is upgraded or if privileged roles are exercised.

- `name()` = `Immunefi`
- `symbol()` = `IMU`
- `decimals()` = `18`
- `paused()` = `false`
- `totalSupply()` = `10,000,000,000` tokens (18 decimals)

5. Verification & Dependency Review

This section summarizes the steps taken to increase confidence that (i) the reviewed sources correspond to the deployed contracts and (ii) the deployed contracts behave as expected. It includes byte-

code reproduction, OpenZeppelin source provenance and advisory triage, mainnet-fork functional tests, a checklist of security-relevant properties reviewed, and key on-chain event history review.

5.1 Bytecode reproduction

To confirm that this review covers the exact deployed code, we recompiled the verified source packages for both the proxy and the `IMU` implementation using the same compiler settings as used for verification, and compared the resulting deployed runtime bytecode to on-chain `eth_getCode`.

Compilation settings were taken from the verified compiler inputs, including:

- Compiler: `solc v0.8.28+commit.7893614a`
- `evmVersion`: `prague`
- Optimizer: disabled
- `viaIR`: `false`
- Metadata `bytecodeHash`: `ipfs`
- Remappings: as per the verified standard-json input (affects the Solidity metadata hash)
- Implementation: exact runtime bytecode match at `0x39EaCeF0Ee562109dF742C989E9D5227d4C17d2e`
- Proxy: exact runtime bytecode match at `0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5` after accounting for deployment-time immutables (the admin value embedded in the proxy runtime code)

This increases confidence that findings in this report correspond to the contracts deployed at the reviewed addresses.

5.2 Third-Party Dependencies: Risk Review

This project's third-party code is almost entirely OpenZeppelin. The main dependency risk questions are:

- Was OpenZeppelin code tampered with (fork edits)?
- Are any known OpenZeppelin vulnerabilities applicable to the exact modules and versions actually used?

Integrity / Supply-chain check

- Method: for each OpenZeppelin source file present in the verified source packages, we compared the file's full contents against upstream OpenZeppelin release tags (by exact text equality / SHA-256).
- Coverage: this includes both the proxy contract's OpenZeppelin sources embedded in its verified package and the implementation contract's OpenZeppelin dependencies.
- Result: all OpenZeppelin source files included in the verified packages match upstream OpenZeppelin release tags exactly (verified by SHA-256 equality checks). Table 1 enumerates one matching upstream tag per file (note that files unchanged across multiple releases may match more than one tag).
- Header marker nuance: `ERC20PausableUpgradeable.sol` includes the comment "last updated v5.1.0". Its content does *not* match upstream `v5.1.0`; instead, it matches upstream OpenZeppelin Contracts Upgradeable `v5.4.0` exactly. For example, the upstream `v5.1.0` version calls `__Pausable_init_unchained()` in `__ERC20Pausable_init()`, while the upstream `v5.4.0` version has an empty `__ERC20Pausable_init()` body, and the verified sources match the latter. The upstream `v5.4.0` file itself retains the same "last updated v5.1.0" comment, which is consistent with an upstream documentation lag rather than a forked modification.

This "last updated" string is an *upstream OpenZeppelin documentation inconsistency* (inherited by this deployment), not a project-specific change. It is non-executable metadata and does not affect compiled bytecode or runtime behavior. The practical impact is version attribution: relying on header markers alone could lead to incorrect advisory triage. This report treats content matching and bytecode reproduction as the source of truth.

In Table 1, `contracts` refers to OpenZeppelin Contracts and `upgradeable` refers to OpenZeppelin Contracts Upgradeable.

Table 1: OpenZeppelin source provenance (content-matched)

Origin	File	Matched upstream tag	Notes
Proxy	access/Ownable.sol	contracts v5.0.0	
Proxy	interfaces/IERC1967.sol	contracts v5.4.0	
Proxy	proxy/ERC1967/ERC1967Proxy.sol	contracts v5.2.0	
Proxy	proxy/ERC1967/ERC1967Utils.sol	contracts v5.4.0	
Proxy	proxy/Proxy.sol	contracts v5.0.0	
Proxy	proxy/beacon/IBeacon.sol	contracts v5.4.0	
Proxy	proxy/transparent/ProxyAdmin.sol	contracts v5.2.0	
Proxy	proxy/transparent/TransparentUpgradeableProxy.sol	contracts v5.2.0	
Proxy	utils/Address.sol	contracts v5.4.0	
Proxy	utils/Context.sol	contracts v5.0.1	
Proxy	utils/Errors.sol	contracts v5.1.0	
Proxy	utils/StorageSlot.sol	contracts v5.1.0	
Implementation	access/AccessControlUpgradeable.sol	upgradeable v5.4.0	
Implementation	access/IAccessControl.sol	contracts v5.4.0	
Implementation	interfaces/IERC5267.sol	contracts v5.4.0	
Implementation	interfaces/draft-IERC6093.sol	contracts v5.4.0	
Implementation	proxy/utils/Initializable.sol	upgradeable v5.3.0	
Implementation	token/ERC20/ERC20Upgradeable.sol	upgradeable v5.4.0	
Implementation	token/ERC20/IERC20.sol	contracts v5.4.0	
Implementation	token/ERC20/extensions/ERC20BurnableUpgradeable.sol	upgradeable v5.0.0	
Implementation	token/ERC20/extensions/ERC20PausableUpgradeable.sol	upgradeable v5.4.0	Header: v5.1.0
Implementation	token/ERC20/extensions/ERC20PermitUpgradeable.sol	upgradeable v5.4.0	
Implementation	token/ERC20/extensions/IERC20Metadata.sol	contracts v5.4.0	
Implementation	token/ERC20/extensions/IERC20Permit.sol	contracts v5.4.0	
Implementation	utils/ContextUpgradeable.sol	upgradeable v5.0.1	
Implementation	utils/NoncesUpgradeable.sol	upgradeable v5.0.0	
Implementation	utils/Panic.sol	contracts v5.1.0	
Implementation	utils/PausableUpgradeable.sol	upgradeable v5.3.0	
Implementation	utils/Strings.sol	contracts v5.4.0	
Implementation	utils/cryptography/ECDSA.sol	contracts v5.1.0	
Implementation	utils/cryptography/EIP712Upgradeable.sol	upgradeable v5.4.0	
Implementation	utils/cryptography/MessageHashUtils.sol	contracts v5.3.0	
Implementation	utils/introspection/ERC165Upgradeable.sol	upgradeable v5.4.0	
Implementation	utils/introspection/IERC165.sol	contracts v5.4.0	
Implementation	utils/math/Math.sol	contracts v5.3.0	
Implementation	utils/math/SafeCast.sol	contracts v5.1.0	
Implementation	utils/math/SignedMath.sol	contracts v5.1.0	

OpenZeppelin security advisories triage

We referenced advisories published on:

- https://github.com/OpenZeppelin/openzeppelin-contracts/security/advisories
- https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/security/advisories

For each advisory listed below, we checked the affected package and version range (GitHub Advisory Database) and whether the affected module exists in the compiled source set for these deployments.

Advisories considered (and applicability)

- GHSA-4h98-2769-gh6h (CVE-2022-35961): ECDSA signature malleability in OpenZeppelin < 4.7.3. Not applicable: this deployment uses OpenZeppelin v5.x ECDSA.sol (exact match to upstream v5.1.0).
- GHSA-mx2q-35m2-x2rh (CVE-2023-30541): TransparentUpgradeableProxy clashing selector calls may not be delegated in OpenZeppelin < 4.8.3. Not applicable: the proxy uses OpenZeppelin v5.x (the deployed TransparentUpgradeableProxy.sol content matches upstream v5.2.0). Note: the transparent proxy "admin can't call implementation" behavior still exists by design.
- GHSA-9vx6-7xxf-x967 (CVE-2024-27094): Base64 encoding may read from potentially dirty memory (affects OpenZeppelin v5 < 5.0.2). Not applicable: no Base64 library is present in the compiled source set for these deployments.
- GHSA-9rcw-c2f9-2j55 (CVE-2025-54070): Bytes.lastIndexOf out-of-bounds access on empty buffers (affects OpenZeppelin >= 5.2.0, < 5.4.0). Not applicable: no Bytes library is present in the compiled source set for these deployments.

- GHSA-5h3x-9wvq-w4m2 (CVE-2023-34234): Governor frontrunning can block proposal creation (< 4.9.1). Not applicable: Governor contracts are not present.
- GHSA-93hq-5wgc-jc82 (CVE-2023-30542): GovernorCompatibilityBravo proposal calldata trimming (< 4.8.3). Not applicable: Governor contracts are not present.
- GHSA-wprv-93r4-jj2p (CVE-2023-34459): MerkleProof multiproof issue (< 4.9.2). Not applicable: MerkleProof is not present.
- GHSA-699g-q6qh-q4v8 (CVE-2023-49798): duplicated execution of subcalls in OpenZeppelin v4.9.4. Not applicable: OpenZeppelin v4.9.4 is not used.
- GHSA-878m-3g6q-594q (CVE-2023-26488): incorrect calculation in OpenZeppelin v4.8.0–v4.8.1. Not applicable: OpenZeppelin v4.x is not used.
- GHSA-g4vp-m682-qqmp (CVE-2023-40014): improper escaping of output (< 4.9.3). Not applicable: OpenZeppelin v4.x is not used.
- GHSA-q4h9-46xg-m3x9: critical UUPSUpgradeable issue (< 4.3.2). Not applicable: UUPSUpgradeable is not present.
- GHSA-vrw4-w73r-6mm8 (CVE-2021-39168): critical TimelockController issue (< 4.3.1). Not applicable: TimelockController is not present.
- GHSA-7j52-6fjp-58gr: ERC2771ContextUpgradeable storage layout inconsistency (< 4.3.0). Not applicable: this module is not present.

Why no additional OpenZeppelin bugs were flagged here

- The system uses a small, widely used surface area of OpenZeppelin (ERC-20 + AccessControl + Pausable + Permit + TransparentUpgradeableProxy).
- OpenZeppelin sources match upstream releases and the resulting bytecode matches on-chain, so the review covers the exact deployed code.
- Published advisories reviewed either target OpenZeppelin v4.x components that are not present, or target specific v5.x modules (Base64, Bytes) that are not present in the compiled source set for these deployments.

Limitations

This does not guarantee the absence of vulnerabilities in OpenZeppelin v5.x; rather, it indicates that no *known published* advisories apply to the exact OpenZeppelin modules compiled into this deployment at the time of review.

5.3 Mainnet-fork testing

We developed and executed a Foundry test suite that interacts with the deployed token via the proxy at 0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5 on an Ethereum mainnet fork pinned to block 24210396.

Tooling & configuration

- Foundry: forge 1.5.1-stable (b0a9dd9ced)
- Fork: Ethereum mainnet at block 24210396

Coverage (feature-level)

The suite validates:

- **Proxy behavior**: EIP-1967 admin/implementation slots match the reported on-chain configuration, and the transparent proxy "admin cannot fallback" restriction is enforced for the ProxyAdmin address.
- **Initialization safety**: initialize(...) cannot be invoked on the proxy (already initialized) and cannot be invoked on the implementation (initializers disabled via _disableInitializers()).
- **ERC-20**: metadata (name, symbol, decimals), transfer/approve/transferFrom success paths and event emission, and standard failure modes (zero addresses, insufficient balance, insufficient allowance). The suite also checks allowance semantics including finite allowance decrease on spend and that type(uint256).max is treated as "infinite approval" and is not decreased.

- **Burnable**: `burn` and `burnFrom` reduce balances and total supply as expected, emit `Transfer` to the zero address, and enforce allowance rules (including infinite allowance behavior).
- **Pausable**: `pause`/`unpause` access control, expected events, correct revert conditions when already paused/unpaused, and `EnforcedPause` semantics on all token movement paths (transfer, mint, burn).
- **AccessControl**: role identifiers, role admin relationships, and role management (`grantRole`/`revokeRole`/`renounceRole`) including expected events and failure modes. The suite validates that privileged actions (`mint`, `pause`, `unpause`) revert for unauthorized callers.
- **Permit (EIP-2612) / EIP-712**: signature approvals via `permit` (success, expiry, wrong signer, replay protection via nonce), and EIP-712 domain correctness including that the verifying contract is the proxy address.
- **Interfaces & invariants**: `supportsInterface` behavior for AccessControl/ ERC165 and basic invariants with fuzz coverage (transfer preserves supply, finite allowance spend accounting, burn/burnFrom supply accounting, and permit while paused).

Results

All tests passed on the forked mainnet state at block 24210396: 47 tests passed (including 6 fuzz tests at 256 runs each), 0 failed.

5.4 Security review checklist (non-exhaustive)

The following checklist summarizes a non-exhaustive set of additional security-relevant properties that were reviewed and/or validated (by code review, on-chain inspection at the reference block, and the mainnet-fork tests described above):

- **Proxy configuration**: EIP-1967 admin and implementation slots match the reported on-chain values at the reference block, and the transparent proxy "admin cannot fallback" rule is enforced (calls from the proxy admin contract revert unless invoking the proxy's upgrade selector).
- **Historical upgrade activity**: the proxy emitted `AdminChanged` and `Upgraded` once each in the deployment transaction; no additional proxy upgrade-related events were observed through the reference block.
- **Upgrade surface**: the implementation contract does not expose upgrade entrypoints (no UUPS-style `upgradeTo`/`upgradeToAndCall` functions); upgrades are mediated by the proxy admin via `upgradeAndCall`.
- **Initialization hardening**: the implementation contract is locked with `_disableInitializers()` and cannot be initialized directly; the proxy is already initialized and rejects re-initialization (`InvalidInitialization`). We also reviewed `Initialized(uint64)` event history to ensure the initializer was not invoked again post-deployment.
- **Custom logic minimization**: the only custom functions are `initialize`, `mint`, `pause`, and `unpause`, and the `_update` override simply delegates to OpenZeppelin without adding fees, blacklists, transfer hooks, or other transfer logic.
- **External-call surface**: token state transitions do not perform external calls to arbitrary recipients/spenders (no ERC-777 style hooks or callbacks), reducing reentrancy and composability risk.
- **Access control boundaries**: privileged actions (`mint`, `pause`, `unpause`, role management) revert for unauthorized callers (`AccessControlUnauthorizedAccount`), and role transitions (`grantRole`/`revokeRole`/`renounceRole`) emit expected events and enforce confirmation requirements (`AccessControlBadConfirmation`).
- **Role administration**: `DEFAULT_ADMIN_ROLE` is the admin of `PAUSER_ROLE` and `MINTER_ROLE` (i.e., it can grant/revoke those roles via `getRoleAdmin`), and the on-chain role holder at the reference block is consistent with the deployment-time `RoleGranted` events.
- **Pausable semantics**: when paused, token movement via the shared `_update` hook is blocked (including transfer, mint, and burn paths), and `pause`/`unpause` are only callable by `PAUSER_ROLE`. The suite validates both revert conditions (`ExpectedPause`/`EnforcedPause`) and event emission. Allowance changes via `approve` / `permit` are not gated by `_update` and remain available while paused (validated in fork tests).
- **ERC-20 invariants**: balance and total supply updates match expected ERC-20 behavior; standard failure modes (zero addresses, insufficient balance/allowance) revert with OpenZeppelin v5 error types, and required events (`Transfer`, `Approval`) are emitted on the standard paths.

- **Allowance semantics**: `transferFrom` and `burnFrom` spend allowance; `type(uint256).max` acts as an "infinite approval" and is not decreased. Consistent with OpenZeppelin v5 behavior, allowance spend accounting remains correct.
- **Permit / EIP-712**: `permit` validates signatures, expires on deadline, is non-replayable via the nonce, and uses an EIP-712 domain separator tied to chain id and the proxy address as the verifying contract (so integrators should use the proxy address for `DOMAIN_SEPARATOR` and signing).
- **Storage layout**: OpenZeppelin v5 upgradeable modules use ERC-7201 namespaced storage slots, reducing the likelihood of storage collisions in upgradeable deployments (while still requiring careful upgrade procedures).

Privileged account type (context)

At the reference block, the `ProxyAdmin.owner()` and all token roles are held by an address that contains deployed bytecode (i.e., it is a contract account, not an EOA). The security of this contract account (e.g., multisig signers, modules, and processes) is operationally important but out of scope for this code-focused review.

5.5 On-chain event history (key event streams)

To complement the code review and fork tests, we reviewed key on-chain event streams emitted by the proxy address `0xb48c6B24f36307c7A1f2a9281E978a9ef2902BA5` from the deployment block (23224867) through the reference block 24210396.

These streams were chosen because they directly affect upgradeability, privileged control, pause usage, and supply changes; this is not an exhaustive index of all on-chain events.

- **Proxy upgrade events**: the proxy emitted one `AdminChanged` and one `Upgraded` event in the deployment transaction (see Section 4.1), and no additional `AdminChanged` or `Upgraded` events were observed through the reference block. The current EIP-1967 slots at 24210396 are consistent with this history.
- **Initialization**: the proxy emitted `Initialized(1)` in the deployment transaction and no additional `Initialized` events were observed thereafter, which is consistent with the initializer guard (and with `initialize` reverting at the reference block in the fork tests).
- **ProxyAdmin ownership**: the `ProxyAdmin` contract at `0xb91D57D8f7BD2898F1251239e5816d2De45Db228` emitted a single `OwnershipTransferred` event in the deployment transaction, setting the owner from `address(0)` to `0x5428915428b4F7e65bdFb18A92b6F274A98334E6`. No additional `OwnershipTransferred` events were observed through the reference block.
- **Role changes**: the token emitted exactly three `RoleGranted` events in the deployment transaction (granting `DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, and `MINTER_ROLE` to `0x5428915428b4F7e65bdFb18A92b6F274A98334E6`), and no `RoleGranted`, `RoleRevoked`, or `RoleAdminChanged` events were observed thereafter. The on-chain role membership at 24210396 matches this history.
- **Pause usage**: no `Paused` or `Unpaused` events were observed through the reference block, indicating that the token has not used its pause mechanism to date (and `paused()` is `false` at 24210396).
- **Supply issuance**: the entire `10,000,000,000` token supply was minted in a single `Transfer` event from the zero address at block 23339400 (transaction `0xbe74c47c85d87aaed7a14c46083a698cc04bb3a7a6393c51cfd1087874170697`), executed by `0x5428915428b4F7e65bdFb18A92b6F274A98334E6` (the `MINTER_ROLE` holder) minting to `0x4038fdbb7541b429779113403828c7361140d6df`. No additional mint events (`Transfer` from the zero address) were observed thereafter, and no burn events (`Transfer` to the zero address) were observed. The sum of minted amounts matches the on-chain `totalSupply()` at the reference block.

6. Severity Definitions

Severity	Definition
Critical	Direct theft of funds or full compromise with no special conditions
High	Loss of funds or major break with realistic conditions / common integrations
Medium	Loss of funds in edge cases, or significant availability / incorrect behavior
Low	Minor risk, limited availability issues, or common compatibility issues
Informational	Best practices, clarity, and monitoring improvements

7. Findings Summary

All findings identified during this review are **Informational** and primarily relate to the system trust model (privileged roles and upgradeability) and integration/operational nuances. No Critical, High, or Medium severity issues were identified in the reviewed code and on-chain configuration at the reference block 24210396.

ID	Severity	Title
F-01	Informational	Centralized controls (roles)
F-02	Informational	Pause semantics are a “global freeze” (transfers + mint + burn)
F-03	Informational	Upgradeable proxy trust model
F-04	Informational	Pause does not block approvals (`approve`/`permit`)

8. Detailed Findings

F-01 (Informational): Centralized controls (roles)

Affected code

- Role setup: `IMU.initialize`
- Privileged actions: `IMU.mint`, `IMU.pause`

Issue

The token includes privileged controls by design:

- `MINTER_ROLE` can mint arbitrary amounts (no supply cap).
- `PAUSER_ROLE` can pause/unpause transfers (and, due to OpenZeppelin semantics, also blocks mint and burn while paused).

This is not a code defect, but it materially affects the system trust model and should be disclosed clearly to token holders and integrators.

On-chain status

At the audit reference block 24210396, `DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, and `MINTER_ROLE` are held by `0x5428915428b4f7e65bdfb18a92b6f274a98334e6`.

Impact

If privileged role holders are compromised or misused, they can:

- mint unbounded supply,
- freeze transfers (and mint/burn),
- grant/revoke roles to change privileged control.

Recommendation

- Document the trust model and the exact privileged capabilities.
- Maintain strong operational controls (multisig, signer hygiene, incident response plan) and monitoring for privileged actions and role changes.

- Consider adding guardrails if desired (e.g., a supply cap or publicly committed governance policies).

F-02 (Informational): Pause semantics are a "global freeze"

Affected code

- OpenZeppelin `ERC20PausableUpgradeable._update` is gated by `whenNotPaused`.
- `IMU._update` delegates to the above behavior

Issue

Because pausing is enforced inside the shared `_update` hook, pausing blocks *transfers, minting, and burning*. This is consistent with OpenZeppelin's design but should be treated as an explicit product decision.

Impact

In an emergency pause, operators also lose the ability to mint or burn (until unpaused). Depending on operational playbooks, this can be desirable or surprising.

Recommendation

- Ensure governance and integrators understand the pause behavior.
- If a transfers-only pause is desired, implement a custom pause gate that does not affect mint/burn paths.

F-03 (Informational): Upgradeable proxy trust model

Affected code

- `TransparentUpgradeableProxy` and its `ProxyAdmin`

Issue

The token is deployed behind a transparent upgradeable proxy. While no upgrades were observed post-deployment through block 24210396, the architecture is upgradeable by design and the upgrade authority can change the token logic in the future. This is not a code defect, but it is a material trust and integration assumption.

Impact

Upgrades can introduce new functionality, change or remove existing behavior, and potentially impact integrations that assume token behavior is immutable.

Recommendation

- Document the upgrade policy and intended guarantees (if any) for token holders and integrators.
- Maintain strong operational controls and monitoring for `Upgraded` / `AdminChanged` events and for new implementation bytecode deployments.
- Consider governance guardrails if desired (e.g., timelocks, staged upgrades, additional review and testing requirements for each upgrade).

F-04 (Informational): Pause does not block approvals (`approve`/`permit`)

Affected code

- OpenZeppelin `ERC20Upgradeable.approve` and `_approve`
- OpenZeppelin `ERC20PermitUpgradeable.permit` and `_approve`
- OpenZeppelin `ERC20PausableUpgradeable._update` (pause gate)

Issue

The pause mechanism is implemented by gating token movement inside the shared `_update` hook. As a result, pausing blocks transfers, minting, and burning, but it does *not* block allowance changes via `approve` or signature-based approvals via `permit`. This is consistent with OpenZeppelin's design but is an important behavioral detail for incident response and integrations.

Impact

During an emergency pause, users (including accounts whose keys are compromised) can still create or modify allowances. While allowance-based transfers cannot execute until unpaused, allowances created during the paused period may be consumed immediately after unpausing. Some integrations may also assume approvals fail while paused.

Recommendation

- Document that `pause` freezes token movement but not approvals (`approve`/`permit`).
- If a "full freeze" is desired, consider adding a pause gate to approval paths (noting this diverges from default OpenZeppelin semantics and may impact integrations).

Disclaimer

Report Date: 2026-01-11

We hope you find this report informative and useful. If you have any questions or need further clarification, please do not hesitate to contact `contact@d23e.ch`.

The report is provided solely for informational purposes and should not be considered as an endorsement, recommendation, or any form of legal, financial, or investment advice.

The report is based on the code at the time and does not account for any updates, modifications, or alterations to the code that may occur after the report date. The code was assessed “as-is” and the findings represent the state of the code at the time of the assessment.

Although every reasonable effort has been made to ensure the accuracy, completeness, and fairness of the report and findings contained within the report, it is provided on an “as-is” basis without any warranties, representations, or guarantees of any kind, express or implied. This includes, but is not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement, accuracy, or the presence or absence of errors, whether or not discoverable.

The authors, evaluators, and any associated parties disclaim all liability for any losses, damages, costs, or expenses (including legal fees) arising directly or indirectly from the use of or reliance on the report or its findings. This includes, but is not limited to, any damage or loss caused by errors, omissions, inaccuracies, or any misleading or out-of-date information.

The reader is solely responsible for any actions or decisions taken based on the information provided in this report. It is highly recommended that, where necessary, appropriate professional advice is sought before making any decisions or taking any actions relating to the smart contract code analyzed in this report.