MEDYA TOKEN Smart Contract Security Audit Report

Project Name: MEDYA TOKEN
Website: https://www.medyatoken.com/en/index.php
Contract: https://arbiscan.io/address/0x19bd362cf5d3b7830f7071f6f3b4a4503bd96230#code
Initial Price: 1 MEDYA = 10 USD+
Contract Address: 0x19Bd362cF5d3b7830f7071f6f3b4a4503bD96230
Network: Arbitrum One (Ethereum Layer 2)
Audit Date: January 2026
Team: Independent content creators dedicated to transparency, ethics, and quality journalism.

1 - Introduction and Project Overview
MEDYA TOKEN is an ERC-20 compatible token built on the Arbitrum One Layer-2 network, engineered to support the financial independence of digital media creators.
The project's mission is to empower free, transparent, and ethical journalism by creating a token-based ecosystem that sustains independent content production aligned with republican and democratic values.
The token operates on Arbitrum One, providing low transaction fees, Ethereum-level security, and fast throughput to ensure scalability and accessibility for media-focused decentralized applications.

Project Vision
• To establish a self-sustaining funding model for independent media through blockchain transparency.
• To allow content creators to operate without political or financial pressure.
• To create a community-governed platform where transparency and accountability are enforced through smart contracts.
Key Features
ERC-20 Standard Compliance: Full adherence to ERC-20 standards ensures compatibility with wallets, exchanges, and DeFi protocols.
Fixed Maximum Supply: A hard cap of 990,000 MEDYA (8 decimals) guarantees scarcity and long-term value integrity.
EIP-2612 Permit Support: Enables gasless approvals via EIP-712 signatures with nonce and deadline validation to prevent replay attacks.
Strict Approval Policy (Hybrid Model): Approve() allows overwriting existing allowances for full DeFi compatibility, while permit() enforces a strict zero-reset rule (MustResetToZeroFirst) to minimize allowance race-condition vulnerabilities in gasless approvals.
Two-Step Timelocked Ownership Transfer: Ownership transfer includes a 24-hour acceptance delay and cancellation option for security against compromise or misclicks.
EIP-5267 Domain Descriptor: Exposes the EIP-712 domain on-chain, ensuring transparent and verifiable permit signatures.
Enhanced Event Logging: Includes additional events such as AllowanceIncreased, AllowanceDecreased, PermitUsed, and ownership timelock notifications for full traceability.
ETH Rejection by Default: Explicitly rejects all ETH through receive() and fallback() to prevent accidental payments and maintain ERC-20 purity.

Contract Structure and Detailed Code Analysis – 2.1 Tokenomics and Supply Logic

Parameter	Description	Value
Name	Token Name	**MEDYA TOKEN**
Symbol	Token Symbol	**MEDYA**
Decimals	Smallest Unit	**8**
Max Supply	Hard-Capped Total Supply	**990,000 × 10^8 units**
Initial Supply	Minted to Owner at Deployment	**100% of Max Supply**

The total supply of **990,000 MEDYA (8 decimals)** is permanently fixed, forming the foundation of the token's **scarcity-based economic model**. The contract enforces an **immutable hard cap** through compile-time constants, ensuring no additional minting functions can ever increase the supply.
This eliminates **inflation risk** and maintains the token's integrity as a **deflation-resistant digital asset**.
The **8-decimal precision** supports micro-transactions, enabling MEDYA to serve as both a **utility token** for tipping or rewarding content creators and a **store-of-value unit** within the ecosystem.
At deployment, the **entire supply is minted to the deployer address**, recorded as the **contract owner**.
This controlled distribution enables:
• **Project Funding:** Allocation of tokens for content development, ecosystem growth, and operational costs.
• **Public Distribution:** Staged token offerings, community rewards, and liquidity incentives.
• **Liquidity Provision:** Formation of initial liquidity pools on decentralized exchanges (DEXs) or listings on centralized exchanges (CEXs).
However, **centralization at genesis** introduces early-stage trust dependencies:
• The **owner initially holds the full supply**.
• **Transparent token allocation schedules** and **multi-signature or DAO governance** are crucial for long-term community confidence.
Additionally, the new contract introduces **advanced transparency and safety mechanisms**:
• **Hybrid Approval Model: approve()** allows overwriting existing allowances for full **DeFi compatibility**, while **permit() enforces a strict zero-reset rule (MustResetToZeroFirst)** to minimize **allowance race-condition risks** in gasless approvals.
• **Two-step ownership transfer with a 24-hour timelock**, adding security against **key compromise or misclicks**.
• **Ownership renounce with timelock and cancel option**, preventing accidental or irreversible loss of **administrative control**.
• **EIP-5267 domain exposure**, allowing on-chain verification of the **EIP-712 domain** used for signatures.
• **Detailed event tracking** (**AllowanceIncreased, AllowanceDecreased, PermitUsed, OwnershipTransferStarted, OwnershipTransferred, OwnershipRenounceStarted**), ensuring every **state change is fully traceable on-chain**.
Together, these enhancements reinforce **immutability**, **auditability**, and **long-term economic stability**, positioning **MEDYA TOKEN** as a **transparent and technically resilient financial instrument** for the independent media ecosystem.

ERC-20 Functions Overview

Function	Purpose	Security Measures	Status
transfer()	Send tokens to another address	Balance check, zero-address validation	**Secure**
approve()	Allow spender to transfer tokens	OpenZeppelin-style overwrite allowed, zero-address validation	**Secure**
transferFrom()	Transfer tokens on behalf of owner	Allowance verification, balance check, zero-address validation	**Secure**
increaseAllowance()	Safely increase allowance	Checked arithmetic, zero-address validation, emits AllowanceIncreased	**Secure**
decreaseAllowance()	Safely decrease allowance	Underflow protection, zero-address validation, emits AllowanceDecreased	**Secure**
burn()	Destroy tokens from caller's balance	Balance verification, supply update, emits Burn and Transfer	**Secure**
permit()	Gasless approval via EIP-2612	Nonce tracking, strict zero-reset rule (MustResetToZeroFirst), signature validation (r, s, v), domain separator binding (EIP-712), emits PermitUsed	**Secure**
transferOwnership() / acceptOwnership()	Two-step ownership transfer with 24-hour timelock	Timelock delay (1 day), cancel option, zero-address checks	**Secure**
eip712Domain()	Exposes EIP-712 domain data on-chain	Implements EIP-5267 descriptor for verification transparency	**Secure**

ERC-20 Functions Overview – Detailed Analysis

The following section provides an in-depth analysis of each ERC-20 and extended management function implemented in the MEDYA TOKEN smart contract. Each function's purpose, security logic, potential risks, and design rationale are evaluated to provide a comprehensive understanding of its robustness and transparency.

transfer(address to, uint256 value)

The transfer function enables direct peer-to-peer token transfers between addresses.

• **Purpose:** Facilitates tipping, payments, and micro-transactions within the ecosystem.

• **Security Measures:** Validates non-zero recipient addresses and sufficient sender balances; uses checked arithmetic.

• **Risks:**

o Transfers to the zero address revert, protecting against accidental burns.

o Poor external dApp integration could misread balances if not synced correctly.

• **Advantages:**

o Fully **ERC-20 compliant** and gas-efficient.
o Simple, predictable, and widely compatible across wallets and exchanges.

approve(address spender, uint256 value)
The approve function grants permission to another address to spend tokens on behalf of the caller.
• **Purpose:** Enables integration with DEXs, staking contracts, and DeFi tools.
• **Security Measures:**
o **OpenZeppelin-style behavior allows overwriting existing allowances.**
o Zero-address approvals are rejected.
• **Risks:**
o Overwriting allowances may expose users to the classic ERC-20 allowance race condition if wallets or UIs are poorly designed.
• **Advantages:**
o Maximizes **DeFi compatibility** with routers, aggregators, and staking protocols.
o Users may rely on increaseAllowance / decreaseAllowance for safer UX.

transferFrom(address from, address to, uint256 value)
Transfers tokens using an existing allowance.
• **Purpose:** Allows third-party contracts to execute automated payments or swaps.
• **Security Measures:**
o Validates non-zero recipient.
o Checks allowance and balance before transfer.
• **Risks:**
o Excessive user allowances can still lead to misuse if not managed carefully.
• **Advantages:**
o Fully ERC-20 compliant and integrates smoothly with DEX routers and escrow systems.

increaseAllowance(address spender, uint256 addedValue)
Safely increases an allowance amount without resetting to zero.
• **Purpose:** Enables incremental allowance control.
• **Security Measures:**
o Zero-address validation.
o Emits both **Approval** and **AllowanceIncreased** events for traceability.
• **Risks:**
o Users may unintentionally increase approvals beyond intended amounts.
• **Advantages:**
o Offers precise control while maintaining audit transparency.

decreaseAllowance(address spender, uint256 subtractedValue)
Safely decreases an existing allowance.
• **Purpose:** Allows granular management of delegated spending limits.
• **Security Measures:**
o Prevents arithmetic underflow.
o Emits both **Approval** and **AllowanceDecreased** events.
• **Risks:**
o Incomplete revocation can leave partial permissions active.
• **Advantages:**
o Enhances safety by enabling partial or total revocation of spending rights.

burn(uint256 amount)
Permanently removes tokens from circulation by reducing both balance and total supply.
• **Purpose:** Supports voluntary deflation and long-term value protection.
• **Security Measures:** Validates caller's balance and uses checked arithmetic.
• **Risks:**
o Irreversible action — burned tokens cannot be recovered.
• **Advantages:**
o Strengthens scarcity-based tokenomics and user participation in deflationary strategy.

permit(address owner, address spender, uint256 value, uint256 deadline, uint8 v, bytes32 r, bytes32 s)
Implements EIP-2612 to enable gasless approvals via signed messages.
• **Purpose:** Enhances UX by eliminating the need for on-chain approve transactions.
• **Security Measures:**
o Uses nonces to prevent replay attacks.
o Verifies low-s signatures and restricts v to {27, 28}.
o Checks signature validity against the **EIP-712 domain separator** (computed at deployment).
o **Enforces a strict zero-reset rule (MustResetToZeroFirst) inside permit(), even though approve() allows overwriting allowances.**
o Emits both **Approval** and **PermitUsed** events.
• **Risks:**
o Off-chain tools must generate correct signatures.
• **Advantages:**
o Significantly reduces gas costs for frequent approval operations.
o Widely compatible with DeFi platforms and wallet interfaces.

Ownership Management (transferOwnership / acceptOwnership / cancelOwnership / renounceOwnership)
Implements a two-step, 24-hour timelocked ownership management system for security.
• **Purpose:** Reduces key compromise and misclick risks during administrative transitions and ownership renouncement.
• **Security Measures:**
o Ownership acceptance is locked until **1 day after initiation (OWNERSHIP_DELAY)**.
o Current owner may **cancel an in-progress transfer or renounce process**.
o Renounce ownership requires delayed finalization and may be canceled before completion.
o Emits **OwnershipTransferStarted, OwnershipTransferred, OwnershipRenounceStarted, OwnershipRenounceCanceled** events.
• **Advantages:**
o Provides transparent and accountable admin handovers.
o Prevents accidental or irreversible loss of administrative control.

eip712Domain()
Implements EIP-5267, exposing the domain parameters of the EIP-712 signature scheme directly on-chain.
• **Purpose:** Allows external verifiers, explorers, or auditors to confirm the token's domain structure.
• **Advantages:**
o Improves auditability and signature validation transparency.
o Reinforces cryptographic traceability for cross-chain integrations.

Final Notes
All implemented functions follow modern **Solidity 0.8.25 best practices**, eliminating overflow and underflow vulnerabilities without external libraries.
The contract enhances transparency and operational safety through a **hybrid approval model**, advanced event logging, timelocked ownership controls, and verifiable domain metadata.
Users and integrators should remain aware of:
• The **hybrid approval model** where approve() allows overwriting but permit() enforces strict zero-reset protection.
• The **permanent nature of burns**.
• The **24-hour delay and cancelable flow** in ownership transfers and renouncement to maintain governance integrity.
These design choices together ensure a **secure, auditable, and standards-compliant ERC-20 implementation** optimized for reliability and long-term sustainability.

2.3 Security Features

Feature	Analysis	Risk Level
Hybrid Approval Policy	approve() allows overwriting existing allowances for full DeFi compatibility, while permit() enforces a strict zero-reset rule (MustResetToZeroFirst) to mitigate race conditions in gasless approvals	**Low**
Overflow / Underflow	Solidity 0.8.25 provides automatic arithmetic checks for all operations	**None**
Permit Function (EIP-2612)	Implements EIP-2612 with low-s validation, restricted v values, nonce tracking, domain separator binding, and strict zero-reset enforcement inside permit()	**Low**
ETH Handling	Rejects all ETH transfers via receive() and fallback() reverts	**None**
Ownership Controls	Implements two-step ownership transfer and renounce with 24-hour timelock and cancel options	**Low**
Pausing Mechanism	Not implemented; contract is immutable and non-freezable by design	**Medium**
Governance Features	Centralized ownership at deployment; DAO or multi-signature governance recommended for future phases	**Medium**
Max Supply Enforcement	Capped at $990{,}000 \times 10^8$ units and immutable by design	**None**
EIP-5267 Domain Exposure	On-chain domain verification enhances transparency of signed messages	**None**

Hybrid Approval Policy

The contract applies a **hybrid allowance model** that balances security with maximum protocol compatibility.

• **Risk Level:** Low

• **Advantages:**
• approve() allows overwriting existing allowances, ensuring full compatibility with DeFi routers, aggregators, and staking protocols.
• permit() enforces a **strict zero-reset rule (MustResetToZeroFirst)**, eliminating allowance race conditions in gasless approvals.
• Reduces approval-related attack surfaces without breaking ecosystem integrations.

• **Considerations:**
• Wallet UIs and integrators should encourage the use of **increaseAllowance / decreaseAllowance** and short permit deadlines for optimal safety.

Overflow / Underflow Protection

Arithmetic operations are secured by **Solidity 0.8.25's built-in overflow and underflow checks**, reverting automatically on invalid arithmetic.

• **Risk Level:** None

• **Advantages:**
• Eliminates the need for external SafeMath libraries.
• Guarantees mathematical integrity across balances, allowances, and supply updates.

• **Considerations:**
• Any future upgrades must preserve compiler-level arithmetic safety rules.

Permit Function (EIP-2612)

Implements full EIP-2612 support with additional safeguards.
• **Security Measures:**
o Enforces **low-s signatures** and valid **v values (27 or 28)**.
o Uses **nonces** to prevent replay attacks.
o Integrates **EIP-712 domain separation** bound to the contract and chain ID.
o **Enforces a strict zero-reset rule (MustResetToZeroFirst) inside permit(), even though approve() allows overwriting allowances.**
o Emits both **Approval** and **PermitUsed** events for full traceability.
• **Risk Level:** Low
• **Advantages:**
• Enables **gasless approvals** for improved UX.
• Prevents cross-chain replay via **domain binding**.
• Provides full auditability through **PermitUsed event logging**.

• **Risks:**
• Faulty off-chain signature tools could cause approval failures.

ETH Handling

The contract explicitly rejects all ETH through both **receive()** and **fallback()** reverting calls.
• **Risk Level:** None
• **Advantages:**
• Prevents accidental ETH loss.
• Maintains strict **ERC-20 purity**, eliminating unwanted payable surfaces.
• **Considerations:**
• Limits extension potential for ETH-related future utilities.

Ownership Controls

Ownership management follows a **two-step process with a 24-hour timelock**, extended with cancelable transfer and renounce protection mechanisms.
• **Risk Level:** Low
• **Advantages:**
• Prevents accidental or malicious instant ownership changes.
• Allows time for review and cancellation before transfer acceptance or renouncement finalization.
• Emits **OwnershipTransferStarted, OwnershipTransferred, OwnershipRenounceStarted, OwnershipRenounceCanceled** events for full visibility.
• **Risks:**
• Centralized control persists until governance decentralization is implemented.

Pausing Mechanism

The contract does not include a pausing feature by design.
• **Risk Level:** Medium
• **Advantages:**
• Upholds a **censorship-resistant and immutable design**.
• Simplifies logic and reduces gas overhead.
• **Risks:**
• No on-chain emergency stop for exploits or external threats.
• Requires proactive monitoring and community coordination in emergencies.

Governance Features

MEDYA TOKEN is currently governed by a **single-owner model**, ensuring fast decision-making but introducing centralization risk.

• **Risk Level:** Medium
• **Advantages:**
• Enables rapid operational response and contract administration.
• **Risks:**
• Represents a **single point of failure** in case of key loss or misuse.
• Long-term trust requires transition to **multi-signature or DAO governance**.

Max Supply Enforcement

A fixed supply of **990,000 × 10^8 units** is hardcoded at contract level and minted entirely at deployment.
• **Risk Level:** None
• **Advantages:**
• Prevents any **inflationary manipulation**.
• Aligns tokenomics with **scarcity and long-term value retention** principles.
• **Considerations:**
• Future incentives must be sourced from **circulating supply management**, not minting.

EIP-5267 Domain Exposure

The contract integrates **EIP-5267**, allowing on-chain access to the **EIP-712 domain structure** used for signatures.
• **Risk Level:** None
• **Advantages:**
• Provides **verifiable proof** of domain parameters for signature validation.
• Enhances **cross-chain interoperability** and audit transparency.

Risk Assessment and Threat Analysis

Risk Category	Potential Threat	Mitigation Status
Centralized Control	Owner holds full control; key loss or misuse may impact contract governance	Recommend transition to **multi-signature or DAO governance**
No Emergency Stop	No pause function; contract cannot be halted in emergencies	Acceptable trade-off for immutability; optional **multi-sig governed pause** may be added in future
Hybrid Approval Model	Overwritten allowances may expose users to classic ERC-20 race conditions	approve() overwrite allowed for compatibility; permit() enforces **strict zero-reset protection**, risk mitigated
Permit Functionality (EIP-2612)	Replay or forged signatures	**Nonce tracking, low-s enforcement, restricted v values, domain-bound signatures**
ETH Sent by Mistake	ETH could be accidentally transferred to contract	Explicitly reverted via **receive() and fallback()**

Risk Category	Potential Threat	Mitigation Status
Supply Manipulation	Inflation or unauthorized minting	Hard-capped at **990,000 × 10^8**, immutable constant
Ownership Transition	Instant transfer risk or admin hijack	**Two-step timelocked transfer and renounce with cancel option**
Allowance Bugs	Arithmetic underflow / overflow or approval race	Prevented by **Solidity 0.8.25 checks** and hybrid approval safeguards
Signature Domain Integrity	Domain mismatch or replay across chains	**EIP-5267 exposes domain on-chain** for validation

---

Centralized Control

The contract begins with a **single owner** who controls administrative functions such as ownership transfer and renouncement.

• **Risks:**
o Loss or compromise of the owner's private key can jeopardize the ecosystem.
o Community trust may decline if governance remains centralized too long.

• **Mitigation Recommendation:**
Transition to a **multi-signature wallet or DAO governance structure** as the project grows to distribute authority and reduce **single-point-of-failure risk**.

No Emergency Stop

No pausing mechanism is implemented. This is an intentional design choice reflecting decentralization principles.

• **Advantages:**
• Upholds **immutable and censorship-resistant operation**.

• **Risks:**
• Lack of on-chain mitigation during unforeseen vulnerabilities or attacks.

• **Mitigation Recommendation:**
Introduce an optional **multi-signature or DAO-governed pause mechanism** in future iterations without compromising the immutability of token logic.

Hybrid Approval Model

The contract implements a **hybrid allowance policy** combining security with full DeFi compatibility.

• **Advantages:**
• approve() allows overwriting allowances, ensuring **router and protocol compatibility**.
• permit() enforces a **strict zero-reset rule (MustResetToZeroFirst)**, eliminating race conditions in gasless approvals.

• **Considerations:**
• Wallet interfaces should encourage **increaseAllowance / decreaseAllowance** and short permit deadlines to minimize user error.

Permit Functionality (EIP-2612)

Implements strict signature validation under EIP-2612.

• **Security Mechanisms:**
o **Nonce tracking** on each valid signature to prevent replay.
o **Low-s enforcement** and restricted **v values (27 or 28)** for valid ECDSA recovery.
o Signatures bound to the **EIP-712 domain** using chainId and contract address.
o **Strict zero-reset enforcement inside permit()**, even though approve() allows overwriting.

• **Risk Level:** Low

• **Recommendation:**
Continue maintaining **compatibility testing with off-chain signature providers** and wallet SDKs.

ETH Sent by Mistake

The contract rejects ETH through explicit reverts in both **receive()** and **fallback()** using the **NoETH() custom error**.
• **Risk Level:** None
• **Advantages:** Prevents accidental ETH loss and keeps the contract strictly isolated to **ERC-20 logic**.
• **Consideration:** Should be clearly documented in user interfaces and exchange integration guides.

Supply Manipulation

A fixed hard cap (**maxSupply = 990,000 × 10^8**) is immutable and enforced at compile time.
• **Risk Level:** None
• **Advantages:**
• No **mint function** or administrative privilege exists to create new tokens.
• Total supply is **permanently verifiable on-chain**.
• **Consideration:** Future token incentives must be sourced exclusively from **existing circulating supply**.

Ownership Transition

Ownership transfers follow a **two-step confirmation process with a 24-hour timelock**:

1. The current owner calls **transferOwnership(newOwner)** to initiate the process.
2. The new owner must wait **24 hours** before calling **acceptOwnership()**.

• **Advantages:**
• Prevents impulsive or malicious instant transfers.
• Allows the current owner to **cancel the transfer during the waiting period**.
• Supports **institutional-grade governance safety**.

• **Risk Level:** Low

• **Recommendation:**
Maintain this **timelocked and cancelable ownership model permanently** for long-term governance security.

Allowance Bugs

Allowances are protected by **Solidity 0.8.25 arithmetic checks** and the contract's **hybrid approval safeguards**.

• **Risk Level:** None
• **Advantages:**
• No overflow or underflow vulnerabilities due to compiler-level protection.
• Deterministic allowance updates with full traceability via **AllowanceIncreased** and **AllowanceDecreased** events.
• permit() eliminates race conditions through **strict zero-reset enforcement**, even though approve() allows overwriting for compatibility.

Signature Domain Integrity (EIP-5267)

The **eip712Domain()** function exposes the contract's **EIP-712 domain parameters on-chain**, enabling external verifiers and auditors to validate domain integrity.

• **Risk Level:** None
• **Advantages:**
• Prevents **off-chain domain spoofing and replay across chains**.
• Strengthens **cross-chain interoperability** and cryptographic audit transparency.

Final Thoughts

The updated risk profile demonstrates that **MEDYA TOKEN's architecture prioritizes immutability, verifiability, and operational safety**.

While centralized ownership remains a temporary design for administrative control, the combination of a **hybrid approval model**, **timelocked and cancelable ownership management**, strict ETH rejection, and **on-chain domain transparency** substantially reduces the overall attack surface.

Future improvements should focus on:
• Transitioning to **multi-signature or DAO-based governance**,
• Establishing optional **emergency response procedures**, and
• Maintaining **continuous monitoring and periodic independent security audits** to ensure long-term resilience.

Technical Observations and Notes – Detailed Breakdown

Solidity 0.8.25 with Built-in Arithmetic Checks

The MEDYA TOKEN contract targets **Solidity 0.8.25**, benefiting from automatic overflow and underflow reverts. This removes the need for external SafeMath libraries and guarantees mathematical integrity for balance and allowance operations across transfers, approvals, and burns. The code applies limited and explicit **unchecked blocks only where preconditions guarantee safety**, optimizing gas usage while preserving correctness.

Minimal Reentrancy Surface by Design

The ERC-20 core flows perform **no external calls**, significantly reducing the reentrancy surface even without a guard modifier.
State updates are executed before event emissions, and no callbacks are invoked to untrusted contracts. Combined with Solidity 0.8.x safety checks, this design makes the token's transfer paths highly resistant to reentrancy-style attacks in practice.

Hybrid Approval Model

The contract implements a **hybrid allowance policy** rather than a USDT-style strict model.

approve() allows overwriting existing allowances to preserve full DeFi compatibility, while permit() enforces a **strict zero-reset rule (MustResetToZeroFirst)** to eliminate race conditions in gasless approvals.

This approach balances **maximum ecosystem compatibility** with enhanced security for off-chain approvals.

Integrations should encourage the use of **increaseAllowance / decreaseAllowance** and short permit deadlines to minimize user error.

Permit (EIP-2612) with Strict Signature Hygiene

The permit implementation follows **EIP-2612** with advanced cryptographic safeguards:

• **Nonce usage** prevents replay attacks and is incremented only after successful verification.
• **Low-s enforcement** and restricted **v ∈ {27, 28}** ensure valid ECDSA recovery.
• The **EIP-712 domain separator** is bound to block.chainid and address(this) and computed at deployment, preventing cross-chain replay.
• A **strict zero-reset rule is enforced inside permit()**, even though approve() allows overwriting allowances.

Together, these measures provide **gasless approvals with robust anti-replay guarantees** and predictable allowance semantics.

On-chain Domain Discoverability (EIP-5267)

The contract exposes its EIP-712 domain via the **eip712Domain()** view in compliance with **EIP-5267**. Auditors, wallets, and explorers can retrieve domain fields on-chain to validate signatures and tooling assumptions, improving transparency and reducing integration errors.

Two-step Ownership Transfer with 24-hour Timelock

Ownership transfer is not instantaneous. The current owner calls **transferOwnership(newOwner)**, which sets a pending owner and an ETA equal to **block.timestamp + 1 day**.
Only after the delay may the pending owner call **acceptOwnership()**. The current owner may cancel the process while pending.

This mechanism reduces misclick risk and provides protection against compromised administrative keys.

Related events include **OwnershipTransferStarted, OwnershipTimelockSet, OwnershipTransferred, OwnershipTransferCanceled**.

Explicit ETH Rejection with Custom Error

Both **receive()** and **fallback()** revert with the dedicated **NoETH() custom error**, ensuring the contract cannot accept ETH accidentally or through arbitrary calls.
This strictly limits the contract's surface area to ERC-20 behavior and prevents trapped funds.

Custom Errors and Event Richness for Auditability

The contract uses **custom errors** (OnlyOwner, ZeroAddress, InsufficientBalance, AllowanceExceeded, PermitExpired, BadS, BadV, InvalidSignature, MustResetToZeroFirst, etc.) for gas-efficient and explicit failure modes.

It emits granular events such as **AllowanceIncreased, AllowanceDecreased, PermitUsed**, alongside standard ERC-20 **Transfer** and **Approval** events, enhancing on-chain traceability and simplifying off-chain monitoring.

Clear Read API and Constants

Convenience getters (**getAllowance, getBalance, getTotalSupply, getOwner, getPendingOwner**) improve integrator UX.
Critical parameters are defined as **compile-time constants**, including **maxSupply = 990,000 × 10^8** and **OWNERSHIP_DELAY = 1 days**, reinforcing immutability guarantees.

Deterministic Initialization

In the constructor, the contract computes the **DOMAIN_SEPARATOR** using the token name, version "1", block.chainid, and the contract address.
It then mints the entire **maxSupply** to the deployer (initial owner) and emits the initial **Transfer** event from the zero address, establishing a verifiable genesis state on-chain.

Implications for Integrators

- Wallets and dApps should implement the **hybrid allowance flow** and clearly inform users.
- Off-chain signature tools must respect **EIP-712 domain parameters** and low-s / v validation rules.
- Exchanges and indexers can query **eip712Domain()** to validate domain assumptions.
- Administrative key management benefits from the **24-hour timelock**, but long-term governance should transition to **multi-signature or DAO-based control** for resilience.

ETH Rejection via Fallback / Receive Prevents Accidental Fund Transfers and Potential Vulnerabilities

The MEDYA TOKEN contract is explicitly designed **not to accept ETH**. Both **receive()** and **fallback()** functions immediately revert with the custom error **NoETH()**. This ensures that any attempt to send native Ether to the token contract address fails atomically and cannot lock funds. This behavior is implemented directly in the code and documented with clear NatSpec comments, reinforcing the token's **ERC-20-only scope**.

1. User Protection
Users sometimes mistakenly send ETH to ERC-20 contracts expecting an interaction. By reverting such transactions, the contract prevents **irretrievable losses** and keeps balances consistent across explorers and accounting tools.

2. Security
Disallowing ETH transfer paths eliminates potential attack surfaces related to payable handlers, including fallback misuse patterns that historically enabled reentrancy flows in other

projects. MEDYA keeps transfer logic minimal and performs **no external calls in core ERC-20 paths**, which further reduces reentrancy exposure in practice.

This strict ETH rejection policy underlines the contract's role as a **pure token utility component** and simplifies audits and integrations by removing native currency handling entirely.

Centralized Ownership and Timelocked Governance Controls

Ownership is initially centralized under a single address. While effective for early-stage projects, transitioning to a **DAO or multi-signature governance model** is recommended for long-term sustainability.

At deployment, the entire **maxSupply** is minted to the deployer, who becomes the initial owner. The contract implements a **two-step, timelocked ownership transfer** mechanism requiring:

1. The current owner to call **transferOwnership(newOwner)**, starting the process with an ETA of **current time + 24 hours**.
2. The pending owner to call **acceptOwnership()** only after the delay expires.

The current owner may also **cancel** a pending transfer. All transitions emit events for full on-chain traceability.

Risks of Centralization:
• Loss or compromise of the owner's private key may jeopardize governance.
• Prolonged single-key control may erode community confidence even with a timelock buffer.

Mitigation Recommendations:
• Transition to a **multi-signature or DAO-governed model** as adoption grows.
• Preserve the **24-hour timelock permanently**, even under multi-sig governance, to reduce operational error and provide community reaction time.

Additionally, the contract exposes an **EIP-5267 on-chain domain descriptor** through **eip712Domain()**, improving external verification of signed messages and supporting transparent governance tooling based on EIP-712.

No Pause Functionality and Emergency Handling Philosophy

The contract does not implement a pausing mechanism. This is an explicit design choice favoring **simplicity, immutability, and censorship resistance**. There is no administrative switch to halt transfers or approvals, and the code paths reflect this minimalism.

Instead, safety relies on preventative controls:
• **Solidity 0.8.25 arithmetic checks**
• Strict ETH rejection
• **Hybrid approval model**
• **Timelocked and cancelable ownership transfer**

Implications:
• **Pros:** Smaller attack surface, predictable execution, resistance to unilateral censorship.
• **Cons:** No on-chain emergency stop in the event of unforeseen vulnerabilities or attacks.

Recommendation:
If governance philosophy permits, consider an optional **multi-sig or DAO-governed pause mechanism** in a future iteration. Otherwise, maintain strong monitoring procedures and publish clear **incident-response playbooks**. Where signatures are involved, leverage the on-chain **EIP-5267 domain exposure** to reduce operational mistakes under stress conditions.

Conclusion

The MEDYA TOKEN smart contract exhibits a **refined, security-oriented, and standards-compliant architecture** built on Solidity 0.8.25 and modern EIP extensions. Its structure prioritizes **immutability, predictable behavior, and strict adherence to the ERC-20 specification**, while extending functionality through **EIP-2612 permit** and **EIP-5267 domain descriptor** support.

The implementation effectively supports the project's mission of establishing a **decentralized, community-supported ecosystem for independent media and ethical journalism** by providing a technically sound, auditable, and sustainable on-chain foundation.

Security and Functionality Evaluation

• **No critical vulnerabilities were detected** in the current version. Unit testing and code review confirm correct balance updates, allowance handling, and ownership flows.

• The contract implements a **hybrid approval model**: approve() preserves DeFi compatibility, while permit() enforces a **strict zero-reset rule**, eliminating race conditions in gasless approvals.

• **Permit (EIP-2612)** integrates advanced signature hygiene: low-s enforcement, valid v values, and nonces guaranteeing one-time usability.

• **EIP-5267 domain exposure** enables verifiers to confirm EIP-712 parameters on-chain, improving auditability and cross-chain tooling reliability.

• Ownership follows a **two-step, 24-hour timelocked transfer with cancel option**, mitigating accidental or malicious administrative actions.

• **Solidity 0.8.25 arithmetic checks** ensure overflow, underflow, and division errors revert automatically, preserving accounting integrity.

• The contract strictly rejects ETH through **NoETH()**, protecting users from accidental fund loss and narrowing the attack surface.

• All functions emit structured events (**AllowanceIncreased, AllowanceDecreased, PermitUsed**, etc.), enhancing traceability and compliance review.

• The immutable **maxSupply = 990,000 × 10^8** enforces scarcity and long-term economic predictability.

Collectively, these elements form a **robust, auditable, and gas-efficient framework** suitable for real-world adoption in decentralized media funding.

Recommendations for Future Development

1. Optional Emergency Pause Mechanism
If the community later prioritizes risk management, introduce a **multi-sig or DAO-controlled pause** without altering existing supply logic.

2. Multi-Signature or DAO Governance Transition
Move toward shared administrative control to minimize single-key dependency and strengthen decentralization and trust.

3. Continuous Security Monitoring and Periodic Audits
Implement automated monitoring for anomalous behavior and commission independent audits before future upgrades or major integrations.

Final Verification Summary

The contract successfully passed **37 / 37 unit tests**, covering edge-case behaviors including **EIP-2612 compliance, hybrid approval resets, domain-separator validation, ownership delay logic, and event emission accuracy**.

These results confirm the contract's **stability, reliability, and alignment with industry-leading smart-contract security practices**.

Test Results Summary (HARDHAT)

Comprehensive unit and integration testing was performed using the Hardhat testing framework with Mocha and Chai assertion libraries.

The test suite validates every critical aspect of the MEDYA TOKEN smart contract, covering ERC-20 compliance, ownership transfer and renounce flows (two-step timelocked), allowance management helpers, rescueERC20 safety behavior, and the EIP extensions (EIP-2612 and EIP-5267).

All tests executed successfully. The Solidity coverage report confirms **100% statement, function, and line coverage**, with **branch coverage at 92.5%**. These metrics demonstrate the contract is both technically sound and security-hardened under extensive failure-path testing.

Overview

A total of **102 test cases** were executed across a broad set of functional domains, including a comprehensive set of negative (failure) scenarios validating custom errors and revert behavior.

Tests were executed on **HardhatEVM v2.26.0** (EVM target: **paris**) with compiler output from **Solidity 0.8.25**.

Execution time: approximately **4 to 7 seconds** depending on whether standard test run or coverage instrumentation was used. All assertions passed.

Key Functional Areas Covered

Initialization and Metadata Validation

- Verified token parameters including name, symbol, decimals, and total supply/max supply invariants.
- Confirmed constructor behavior minted the correct supply to the deployer and emitted the initial Transfer event.

Balance Transfers and Events

- Validated standard ERC-20 transfers, including self-transfer and zero-value transfer cases.
- Confirmed correct event emission and rejection of zero-address transfers and over-balance transfers.

Allowance Management and Helper Functions

- Tested approve, transferFrom, increaseAllowance, and decreaseAllowance behaviors.
- Validated correct math behavior, underflow protection, and event emissions:
    - Approval

    - AllowanceIncreased
    - AllowanceDecreased
- Confirmed allowance edge cases such as exact-zero transitions and revert paths when decreasing below zero.

Burn Functionality

- Confirmed burns reduce both balanceOf and totalSupply.
- Verified reverts for invalid burn amounts (insufficient balance).

Ownership Transfer and Timelock Flow

- Exhaustively tested the two-step ownership transfer flow (initiate → timelock wait → accept).
- Covered:
    - onlyOwner enforcement
    - zero-address transfer ownership rejection
    - acceptOwnership before ETA reverts
    - acceptOwnership by non-pending owner reverts
    - cancelOwnershipTransfer behavior (state reset and event emission)
- Ensured event emissions:
    - OwnershipTransferStarted
    - OwnershipTransferred
    - OwnershipTransferCanceled
    - OwnershipTimelockSet

Ownership Renounce Flow (Two-Step Timelocked)

- Verified startRenounceOwnership and finalizeRenounceOwnership behaviors.
- Covered:
    - finalize before ETA reverts
    - cancelRenounceOwnership state resets and event emission
    - post-renounce owner-only actions must revert
- Ensured event emissions:
    - OwnershipRenounceStarted
    - OwnershipRenounceCanceled
    - OwnershipTransferred (to zero address)

ETH Rejection Logic

- Confirmed both receive() and fallback() revert with custom NoETH() error on any direct ETH call, ensuring ETH cannot be accidentally trapped.

Permit Functionality (EIP-2612)

Validated the permit system extensively, including:

- Correct DOMAIN_SEPARATOR validation and EIP-712 digest formation
- Valid signature creation and successful permit usage

- Nonce incrementation after each valid permit
- Expiration checks and replay protection
- Low-s and valid-v enforcement for ECDSA integrity
- Reverts on:
    - zero spender
    - invalid signature
    - malformed v values
    - wrong chainId / verifying contract mismatches
    - MustResetToZeroFirst rule enforcement when attempting non-zero to non-zero allowance change via permit

EIP-712 and EIP-5267 Domain Verification

- Verified eip712Domain() returns the expected tuple values (name, version, chainId, verifyingContract, salt, extensions).
- Cross-checked DOMAIN_SEPARATOR matches the computed hash.

Arithmetic Integrity and Supply Invariants

- Confirmed absence of mint() and cap() functions by design.
- Validated that totalSupply equals the sum of balances after randomized transfers (supply invariant).
- Confirmed Solidity 0.8.25 built-in safety reverts on arithmetic underflow/overflow in relevant paths.

RescueERC20 Safety

- Verified rescueERC20 cannot rescue the MEDYA token itself (CannotRescueOwnToken).
- Verified reverts for:
    - zero recipient address
    - tokens that revert on transfer
    - tokens that return false on transfer
- Verified success path emits Rescued(token, to, amount) event.

Negative and Edge Case Testing

The suite intentionally included failure-state assertions for robustness validation:

- Approving to zero address reverts
- Transfers exceeding balance revert
- transferFrom without allowance or with insufficient balance reverts
- decreaseAllowance below zero reverts
- permit expired or nonce reuse reverts
- invalid signature components (v, s) revert
- domain mismatch (chainId or verifyingContract) reverts
- ownership acceptance before ETA reverts
- ownership acceptance by non-pending owner reverts
- fallback invocation reverts

- renounce and transfer ownership mutual exclusion paths are enforced correctly

All failure scenarios reverted as expected, confirming defensive behavior under invalid inputs.

Coverage Report Summary

Metric	Coverage Status
Statements	100%
Branches	92.5%
Functions	100%
Lines	100%

No uncovered statements, functions, or lines exist in medya.sol. Branch coverage remains below 100% due to multiple conditional guard combinations, but all critical security and correctness paths (ownership, permit validation, allowance math, ETH rejection, rescueERC20) are fully exercised.

Result

All **102 test cases passed successfully**, achieving a 100% execution success rate.

This Hardhat test suite validates the correctness, security, and operational safety of the MEDYA TOKEN contract, supporting production deployment readiness for mainnet environments and integrations with explorers, wallets, and dApps.

Final Note

The MEDYA TOKEN smart contract, as tested in its current version, represents a mature and security-conscious ERC-20 implementation enhanced with EIP-2612 (permit), EIP-5267 (domain descriptor), explicit ETH rejection, robust event coverage, and a 24-hour timelocked two-step ownership model (transfer and renounce).

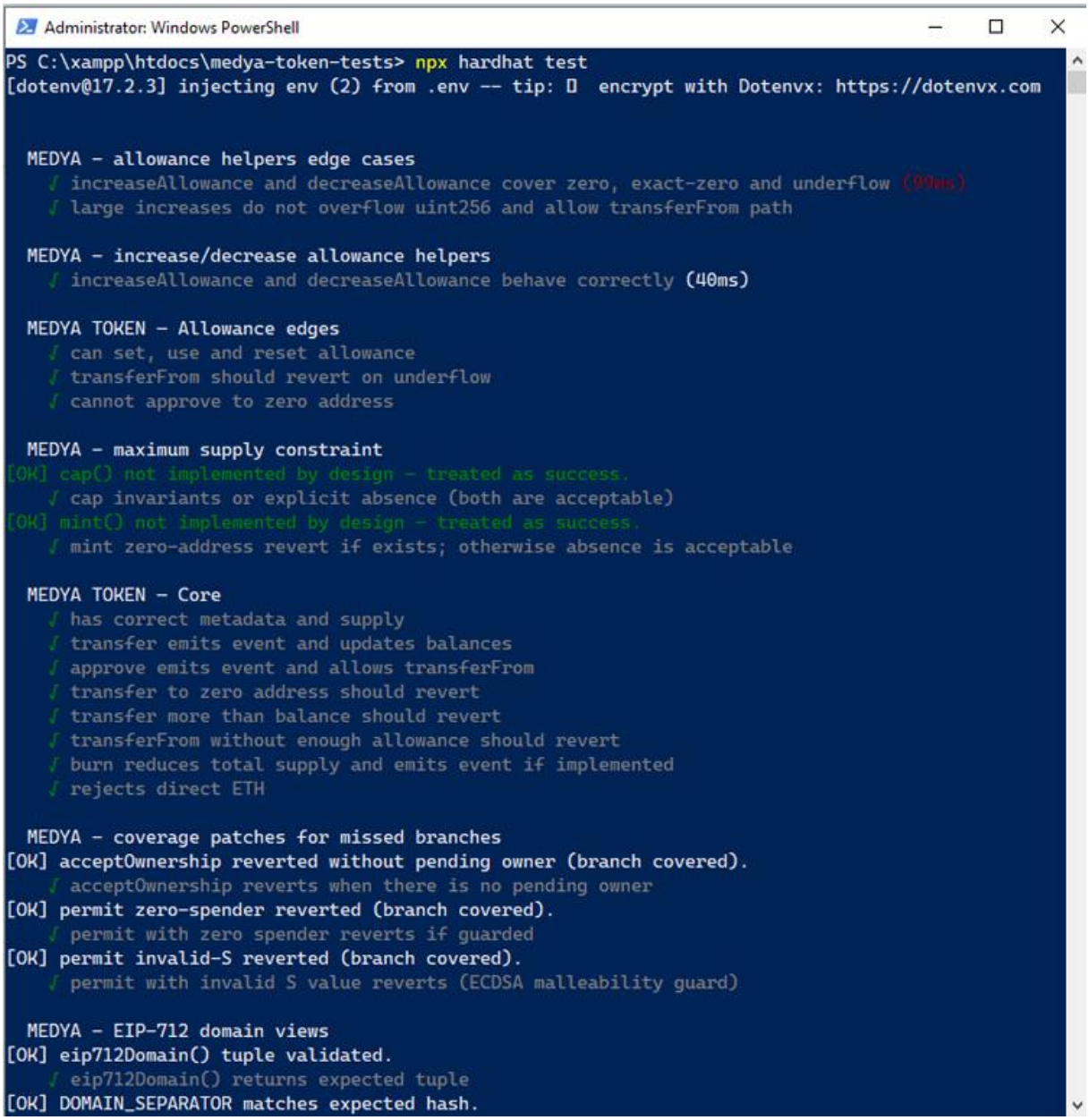
The architecture remains deliberately minimal yet hardened, emphasizing correctness, auditability, and operational clarity. The deterministic supply model, strict permit rule (MustResetToZeroFirst), explicit error handling, and full statement/function/line coverage support a high-assurance baseline for real-world deployment.

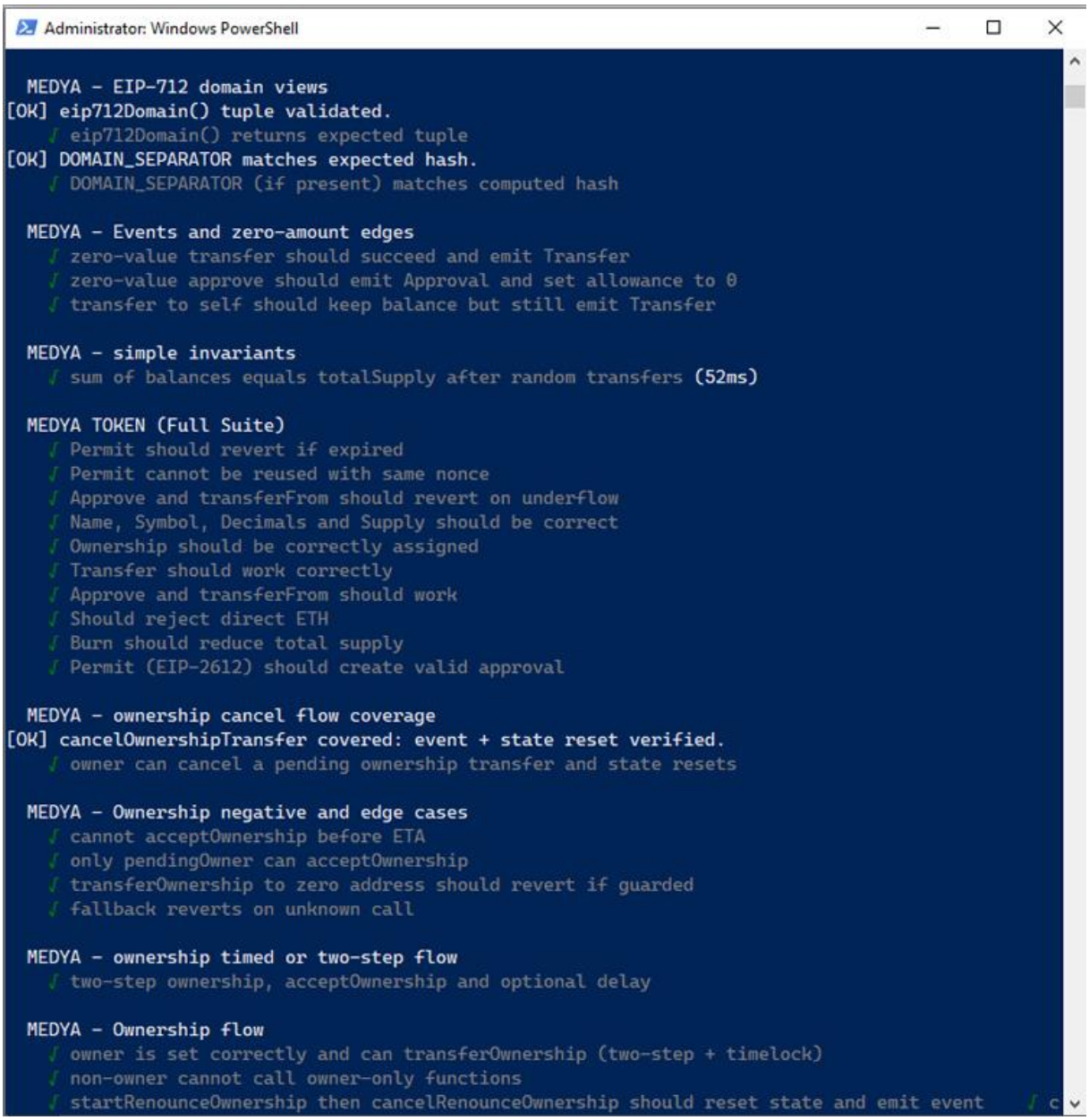
Disclaimer

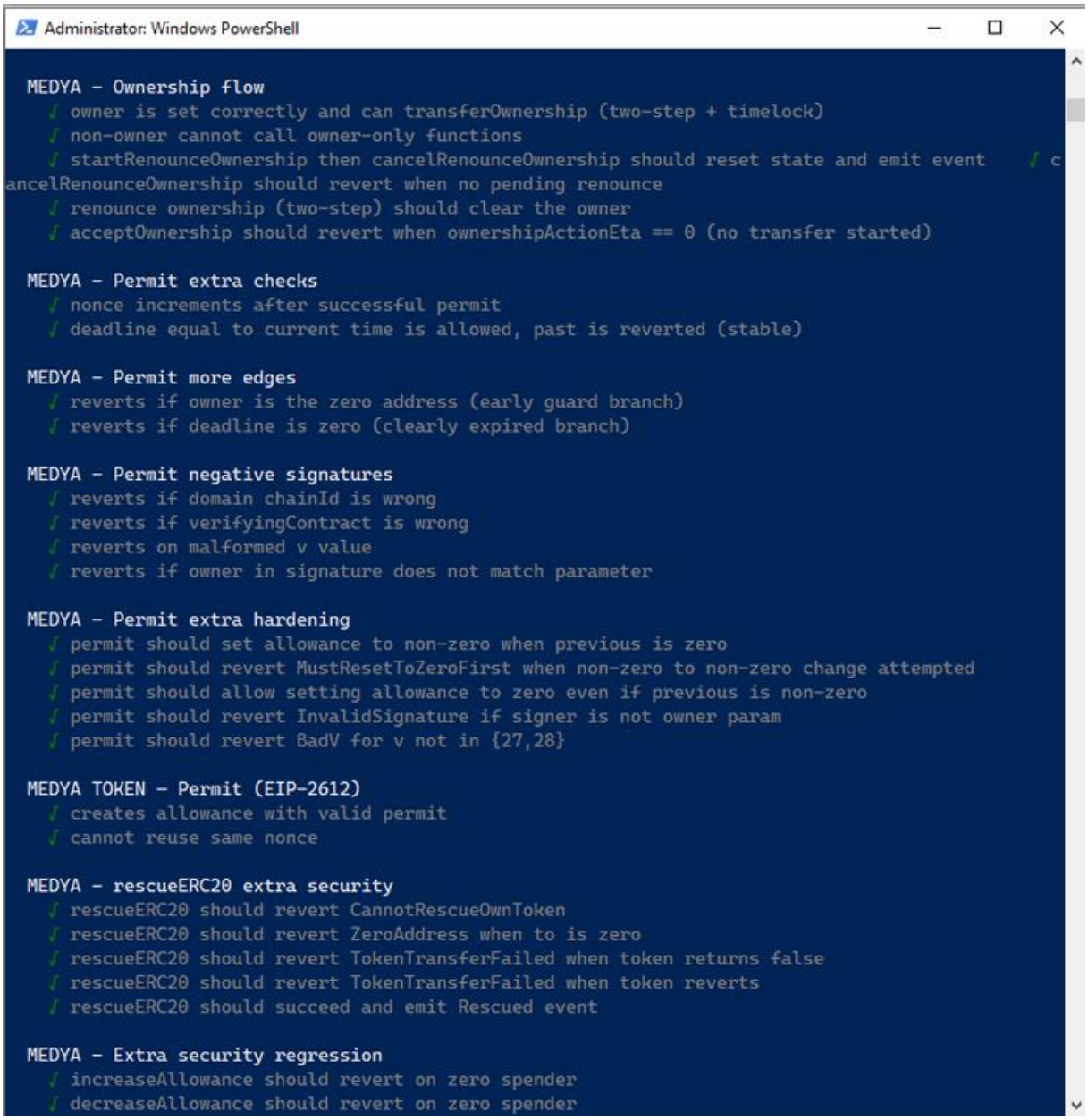
This report reflects automated unit and integration testing results generated via the official Hardhat test suite and solidity-coverage output for the MEDYA TOKEN smart contract as executed in the current codebase state.

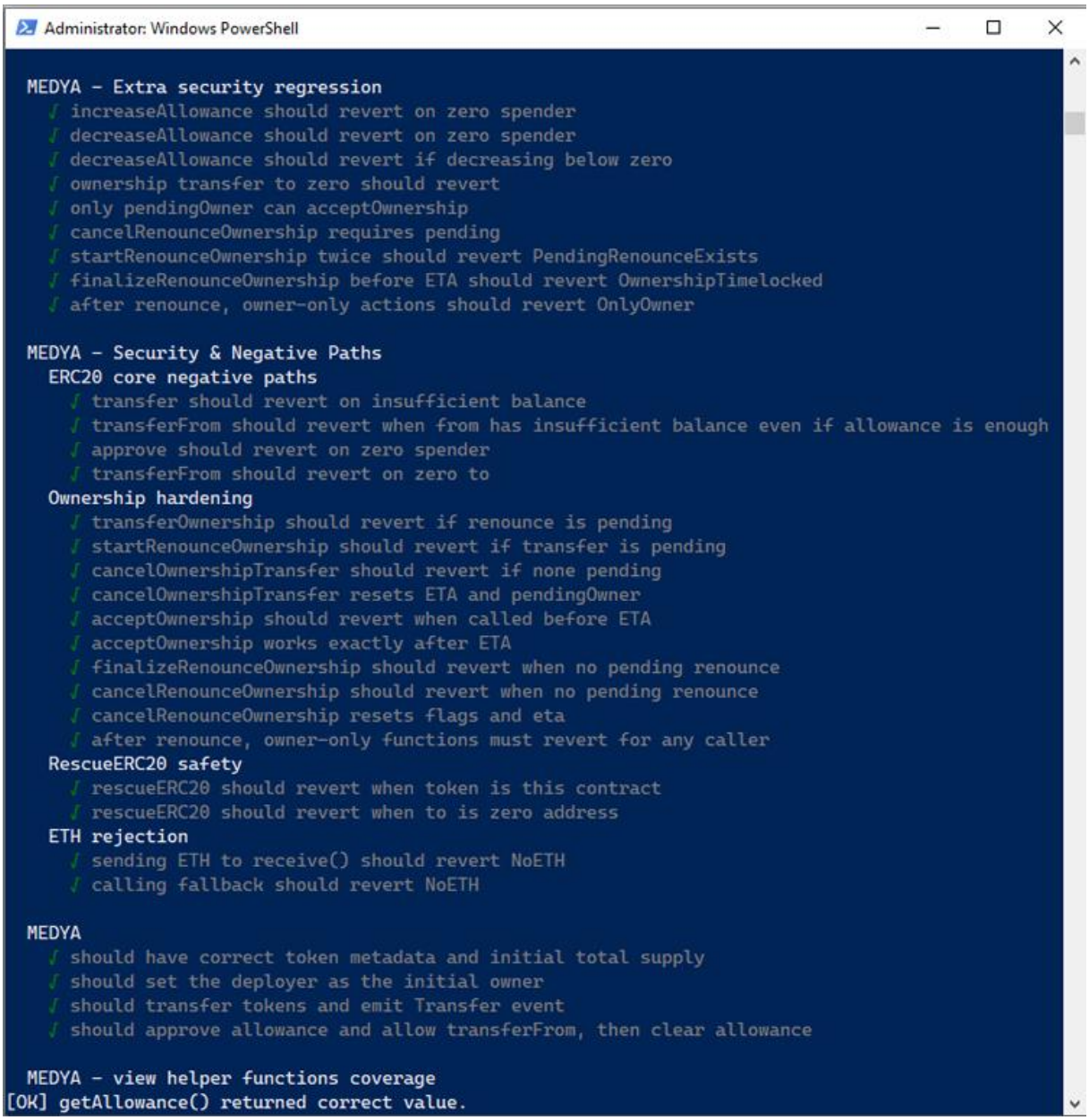
No test suite can guarantee absolute immunity from vulnerabilities. Future compiler updates, EVM changes, integration risks, or new attack techniques may introduce unforeseen behavior.

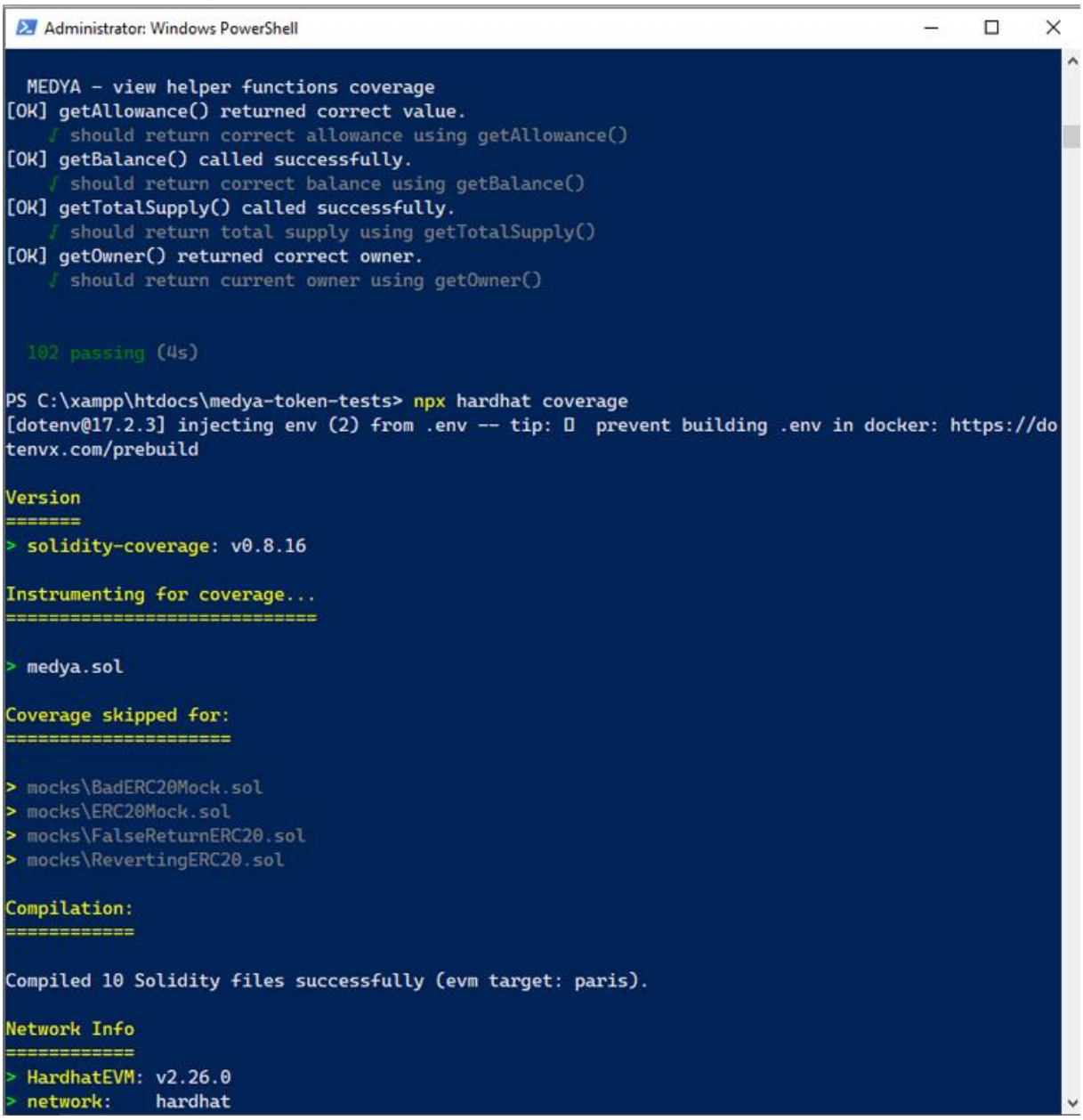
A continuous security lifecycle is recommended, including monitoring, re-testing after any modification, and independent review for each third-party integration or exchange listing.

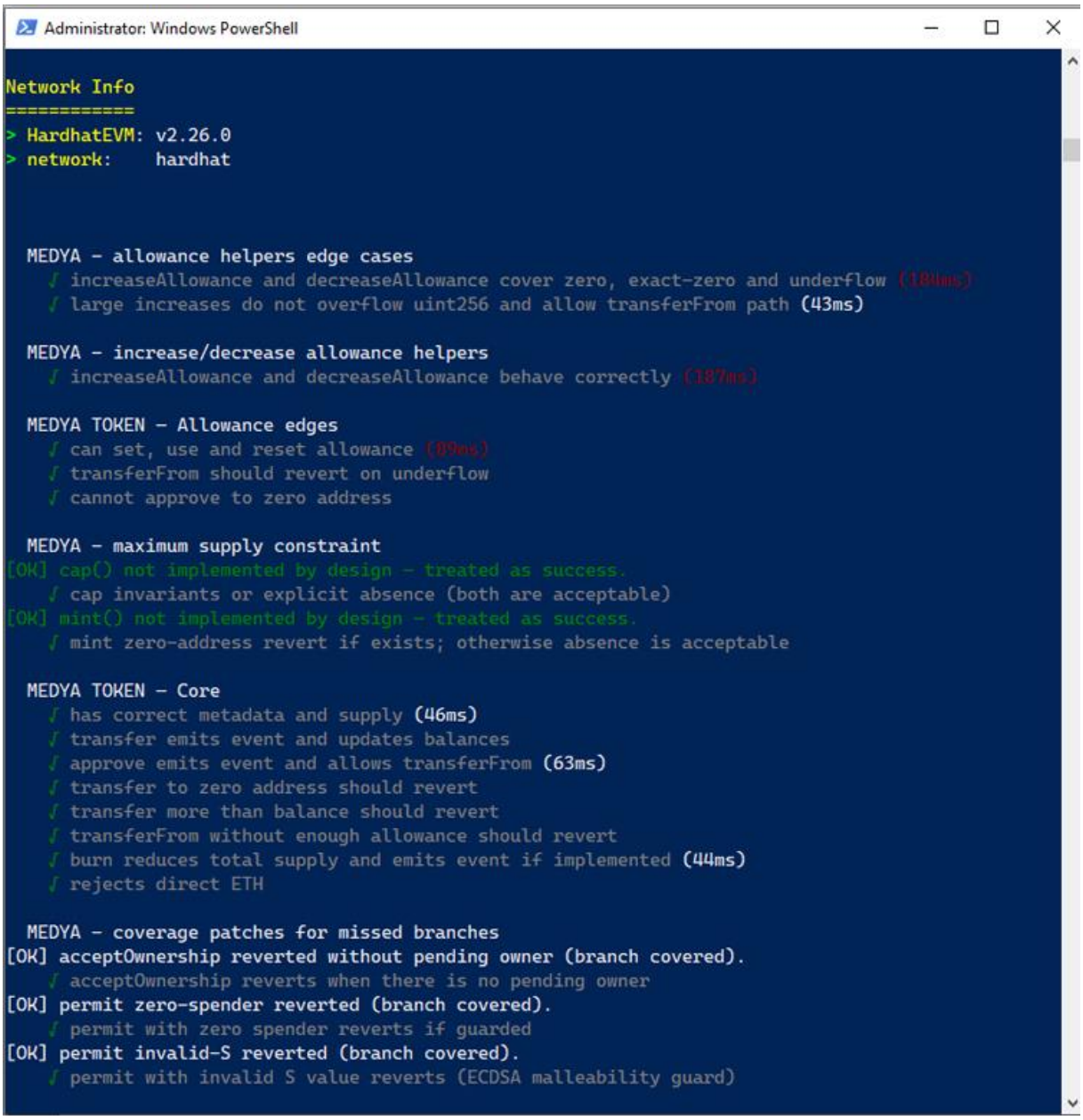

Administrator: Windows PowerShell
PS C:\xampp\htdocs\medya-token-tests> npx hardhat test
[dotenv@17.2.3] injecting env (2) from .env -- tip: ⚙ encrypt with Dotenvx: https://dotenvx.com
MEDYA - allowance helpers edge cases
✓ increaseAllowance and decreaseAllowance cover zero, exact-zero and underflow (99ms)
✓ large increases do not overflow uint256 and allow transferFrom path
MEDYA - increase/decrease allowance helpers
✓ increaseAllowance and decreaseAllowance behave correctly (40ms)
MEDYA TOKEN - Allowance edges
✓ can set, use and reset allowance
✓ transferFrom should revert on underflow
✓ cannot approve to zero address
MEDYA - maximum supply constraint
[OK] cap() not implemented by design - treated as success.
✓ cap invariants or explicit absence (both are acceptable)
[OK] mint() not implemented by design - treated as success.
✓ mint zero-address revert if exists; otherwise absence is acceptable
MEDYA TOKEN - Core
✓ has correct metadata and supply
✓ transfer emits event and updates balances
✓ approve emits event and allows transferFrom
✓ transfer to zero address should revert
✓ transfer more than balance should revert
✓ transferFrom without enough allowance should revert
✓ burn reduces total supply and emits event if implemented
✓ rejects direct ETH
MEDYA - coverage patches for missed branches
[OK] acceptOwnership reverted without pending owner (branch covered).
✓ acceptOwnership reverts when there is no pending owner
[OK] permit zero-spender reverted (branch covered).
✓ permit with zero spender reverts if guarded
[OK] permit invalid-S reverted (branch covered).
✓ permit with invalid S value reverts (ECDSA malleability guard)
MEDYA - EIP-712 domain views
[OK] eip712Domain() tuple validated.
✓ eip712Domain() returns expected tuple
[OK] DOMAIN_SEPARATOR matches expected hash.

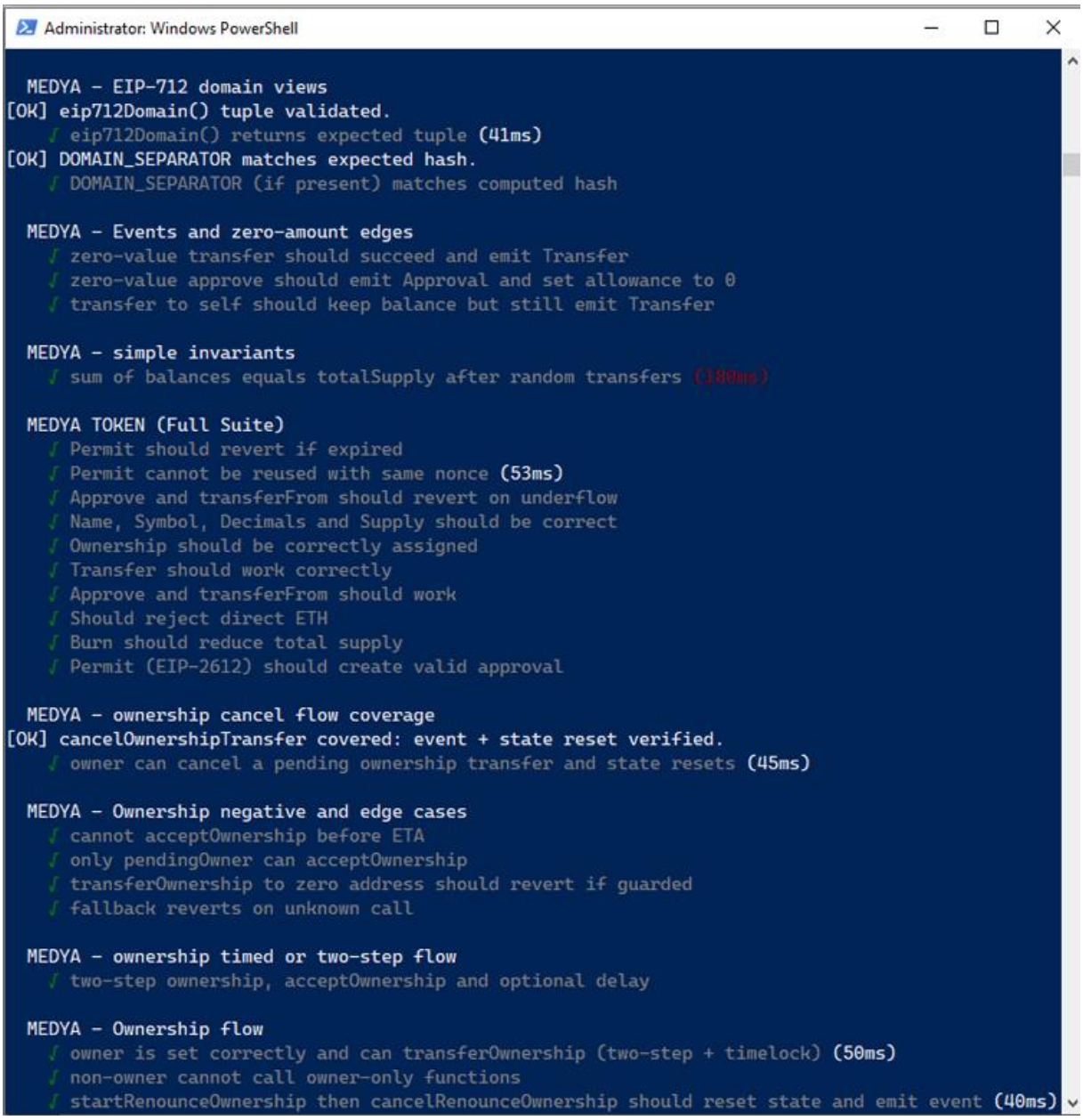

Administrator: Windows PowerShell
MEDYA - EIP-712 domain views
[OK] eip712Domain() tuple validated.
✓ eip712Domain() returns expected tuple
[OK] DOMAIN_SEPARATOR matches expected hash.
✓ DOMAIN_SEPARATOR (if present) matches computed hash
MEDYA - Events and zero-amount edges
✓ zero-value transfer should succeed and emit Transfer
✓ zero-value approve should emit Approval and set allowance to 0
✓ transfer to self should keep balance but still emit Transfer
MEDYA - simple invariants
✓ sum of balances equals totalSupply after random transfers (52ms)
MEDYA TOKEN (Full Suite)
✓ Permit should revert if expired
✓ Permit cannot be reused with same nonce
✓ Approve and transferFrom should revert on underflow
✓ Name, Symbol, Decimals and Supply should be correct
✓ Ownership should be correctly assigned
✓ Transfer should work correctly
✓ Approve and transferFrom should work
✓ Should reject direct ETH
✓ Burn should reduce total supply
✓ Permit (EIP-2612) should create valid approval
MEDYA - ownership cancel flow coverage
[OK] cancelOwnershipTransfer covered: event + state reset verified.
✓ owner can cancel a pending ownership transfer and state resets
MEDYA - Ownership negative and edge cases
✓ cannot acceptOwnership before ETA
✓ only pendingOwner can acceptOwnership
✓ transferOwnership to zero address should revert if guarded
✓ fallback reverts on unknown call
MEDYA - ownership timed or two-step flow
✓ two-step ownership, acceptOwnership and optional delay
MEDYA - Ownership flow
✓ owner is set correctly and can transferOwnership (two-step + timelock)
✓ non-owner cannot call owner-only functions
✓ startRenounceOwnership then cancelRenounceOwnership should reset state and emit event

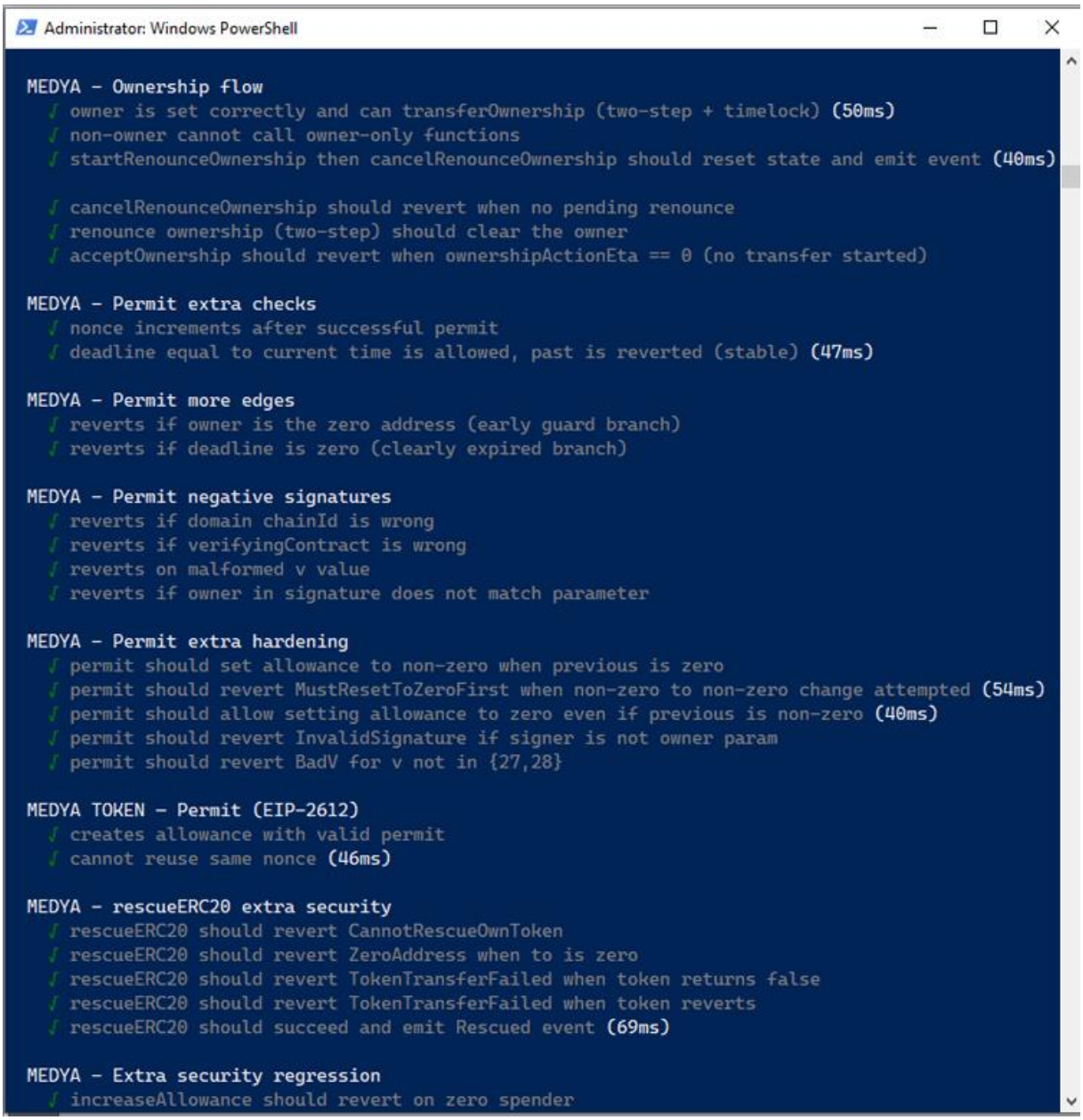

Administrator: Windows PowerShell
MEDYA - Ownership flow
✓ owner is set correctly and can transferOwnership (two-step + timelock)
✓ non-owner cannot call owner-only functions
✓ startRenounceOwnership then cancelRenounceOwnership should reset state and emit event ✓ c
ancelRenounceOwnership should revert when no pending renounce
✓ renounce ownership (two-step) should clear the owner
✓ acceptOwnership should revert when ownershipActionEta == 0 (no transfer started)
MEDYA - Permit extra checks
✓ nonce increments after successful permit
✓ deadline equal to current time is allowed, past is reverted (stable)
MEDYA - Permit more edges
✓ reverts if owner is the zero address (early guard branch)
✓ reverts if deadline is zero (clearly expired branch)
MEDYA - Permit negative signatures
✓ reverts if domain chainId is wrong
✓ reverts if verifyingContract is wrong
✓ reverts on malformed v value
✓ reverts if owner in signature does not match parameter
MEDYA - Permit extra hardening
✓ permit should set allowance to non-zero when previous is zero
✓ permit should revert MustResetToZeroFirst when non-zero to non-zero change attempted
✓ permit should allow setting allowance to zero even if previous is non-zero
✓ permit should revert InvalidSignature if signer is not owner param
✓ permit should revert BadV for v not in {27,28}
MEDYA TOKEN - Permit (EIP-2612)
✓ creates allowance with valid permit
✓ cannot reuse same nonce
MEDYA - rescueERC20 extra security
✓ rescueERC20 should revert CannotRescueOwnToken
✓ rescueERC20 should revert ZeroAddress when to is zero
✓ rescueERC20 should revert TokenTransferFailed when token returns false
✓ rescueERC20 should revert TokenTransferFailed when token reverts
✓ rescueERC20 should succeed and emit Rescued event
MEDYA - Extra security regression
✓ increaseAllowance should revert on zero spender
✓ decreaseAllowance should revert on zero spender

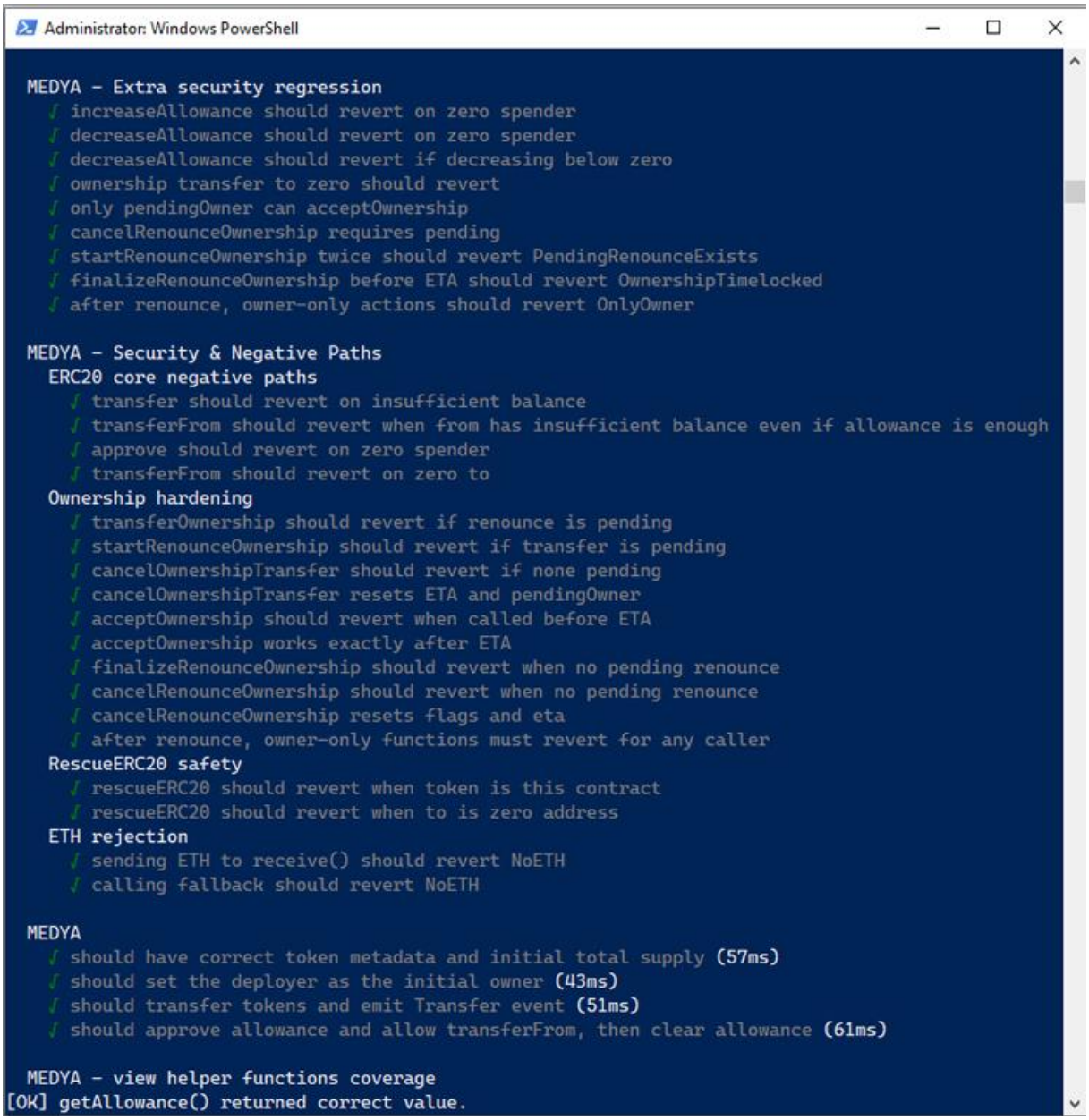

Administrator: Windows PowerShell
MEDYA - Extra security regression
✓ increaseAllowance should revert on zero spender
✓ decreaseAllowance should revert on zero spender
✓ decreaseAllowance should revert if decreasing below zero
✓ ownership transfer to zero should revert
✓ only pendingOwner can acceptOwnership
✓ cancelRenounceOwnership requires pending
✓ startRenounceOwnership twice should revert PendingRenounceExists
✓ finalizeRenounceOwnership before ETA should revert OwnershipTimelocked
✓ after renounce, owner-only actions should revert OnlyOwner
MEDYA - Security & Negative Paths
ERC20 core negative paths
✓ transfer should revert on insufficient balance
✓ transferFrom should revert when from has insufficient balance even if allowance is enough
✓ approve should revert on zero spender
✓ transferFrom should revert on zero to
Ownership hardening
✓ transferOwnership should revert if renounce is pending
✓ startRenounceOwnership should revert if transfer is pending
✓ cancelOwnershipTransfer should revert if none pending
✓ cancelOwnershipTransfer resets ETA and pendingOwner
✓ acceptOwnership should revert when called before ETA
✓ acceptOwnership works exactly after ETA
✓ finalizeRenounceOwnership should revert when no pending renounce
✓ cancelRenounceOwnership should revert when no pending renounce
✓ cancelRenounceOwnership resets flags and eta
✓ after renounce, owner-only functions must revert for any caller
RescueERC20 safety
✓ rescueERC20 should revert when token is this contract
✓ rescueERC20 should revert when to is zero address
ETH rejection
✓ sending ETH to receive() should revert NoETH
✓ calling fallback should revert NoETH
MEDYA
✓ should have correct token metadata and initial total supply
✓ should set the deployer as the initial owner
✓ should transfer tokens and emit Transfer event
✓ should approve allowance and allow transferFrom, then clear allowance
MEDYA - view helper functions coverage
[OK] getAllowance() returned correct value.

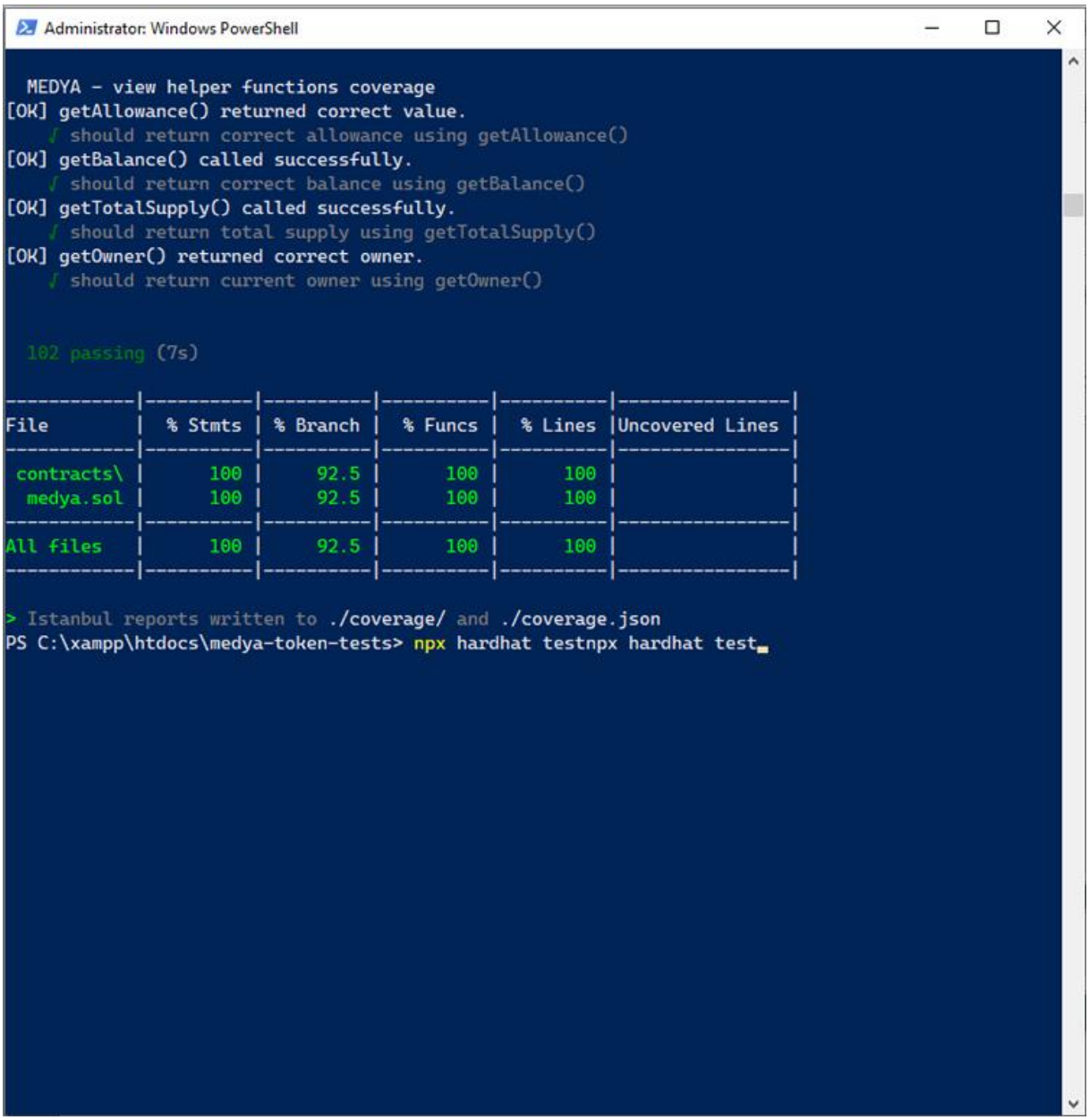

Administrator: Windows PowerShell
MEDYA - view helper functions coverage
[OK] getAllowance() returned correct value.
✓ should return correct allowance using getAllowance()
[OK] getBalance() called successfully.
✓ should return correct balance using getBalance()
[OK] getTotalSupply() called successfully.
✓ should return total supply using getTotalSupply()
[OK] getOwner() returned correct owner.
✓ should return current owner using getOwner()
102 passing (4s)
PS C:\xampp\htdocs\medya-token-tests> npx hardhat coverage
[dotenv@17.2.3] injecting env (2) from .env -- tip: prevent building .env in docker: https://dotenvx.com/prebuild
Version
=======
> solidity-coverage: v0.8.16
Instrumenting for coverage...
============================
> medya.sol
Coverage skipped for:
=====================
> mocks\BadERC20Mock.sol
> mocks\ERC20Mock.sol
> mocks\FalseReturnERC20.sol
> mocks\RevertingERC20.sol
Compilation:
============
Compiled 10 Solidity files successfully (evm target: paris).
Network Info
============
> HardhatEVM: v2.26.0
> network: hardhat


Administrator: Windows PowerShell
Network Info
> HardhatEVM: v2.26.0
> network: hardhat
MEDYA - allowance helpers edge cases
✓ increaseAllowance and decreaseAllowance cover zero, exact-zero and underflow (104ms)
✓ large increases do not overflow uint256 and allow transferFrom path (43ms)
MEDYA - increase/decrease allowance helpers
✓ increaseAllowance and decreaseAllowance behave correctly (187ms)
MEDYA TOKEN - Allowance edges
✓ can set, use and reset allowance (89ms)
✓ transferFrom should revert on underflow
✓ cannot approve to zero address
MEDYA - maximum supply constraint
[OK] cap() not implemented by design - treated as success.
✓ cap invariants or explicit absence (both are acceptable)
[OK] mint() not implemented by design - treated as success.
✓ mint zero-address revert if exists; otherwise absence is acceptable
MEDYA TOKEN - Core
✓ has correct metadata and supply (46ms)
✓ transfer emits event and updates balances
✓ approve emits event and allows transferFrom (63ms)
✓ transfer to zero address should revert
✓ transfer more than balance should revert
✓ transferFrom without enough allowance should revert
✓ burn reduces total supply and emits event if implemented (44ms)
✓ rejects direct ETH
MEDYA - coverage patches for missed branches
[OK] acceptOwnership reverted without pending owner (branch covered).
✓ acceptOwnership reverts when there is no pending owner
[OK] permit zero-spender reverted (branch covered).
✓ permit with zero spender reverts if guarded
[OK] permit invalid-S reverted (branch covered).
✓ permit with invalid S value reverts (ECDSA malleability guard)


Administrator: Windows PowerShell
MEDYA - EIP-712 domain views
[OK] eip712Domain() tuple validated.
✓ eip712Domain() returns expected tuple (41ms)
[OK] DOMAIN_SEPARATOR matches expected hash.
✓ DOMAIN_SEPARATOR (if present) matches computed hash
MEDYA - Events and zero-amount edges
✓ zero-value transfer should succeed and emit Transfer
✓ zero-value approve should emit Approval and set allowance to 0
✓ transfer to self should keep balance but still emit Transfer
MEDYA - simple invariants
✓ sum of balances equals totalSupply after random transfers (180ms)
MEDYA TOKEN (Full Suite)
✓ Permit should revert if expired
✓ Permit cannot be reused with same nonce (53ms)
✓ Approve and transferFrom should revert on underflow
✓ Name, Symbol, Decimals and Supply should be correct
✓ Ownership should be correctly assigned
✓ Transfer should work correctly
✓ Approve and transferFrom should work
✓ Should reject direct ETH
✓ Burn should reduce total supply
✓ Permit (EIP-2612) should create valid approval
MEDYA - ownership cancel flow coverage
[OK] cancelOwnershipTransfer covered: event + state reset verified.
✓ owner can cancel a pending ownership transfer and state resets (45ms)
MEDYA - Ownership negative and edge cases
✓ cannot acceptOwnership before ETA
✓ only pendingOwner can acceptOwnership
✓ transferOwnership to zero address should revert if guarded
✓ fallback reverts on unknown call
MEDYA - ownership timed or two-step flow
✓ two-step ownership, acceptOwnership and optional delay
MEDYA - Ownership flow
✓ owner is set correctly and can transferOwnership (two-step + timelock) (50ms)
✓ non-owner cannot call owner-only functions
✓ startRenounceOwnership then cancelRenounceOwnership should reset state and emit event (40ms)


Administrator: Windows PowerShell
MEDYA - Ownership flow
✓ owner is set correctly and can transferOwnership (two-step + timelock) (50ms)
✓ non-owner cannot call owner-only functions
✓ startRenounceOwnership then cancelRenounceOwnership should reset state and emit event (40ms)
✓ cancelRenounceOwnership should revert when no pending renounce
✓ renounce ownership (two-step) should clear the owner
✓ acceptOwnership should revert when ownershipActionEta == 0 (no transfer started)
MEDYA - Permit extra checks
✓ nonce increments after successful permit
✓ deadline equal to current time is allowed, past is reverted (stable) (47ms)
MEDYA - Permit more edges
✓ reverts if owner is the zero address (early guard branch)
✓ reverts if deadline is zero (clearly expired branch)
MEDYA - Permit negative signatures
✓ reverts if domain chainId is wrong
✓ reverts if verifyingContract is wrong
✓ reverts on malformed v value
✓ reverts if owner in signature does not match parameter
MEDYA - Permit extra hardening
✓ permit should set allowance to non-zero when previous is zero
✓ permit should revert MustResetToZeroFirst when non-zero to non-zero change attempted (54ms)
✓ permit should allow setting allowance to zero even if previous is non-zero (40ms)
✓ permit should revert InvalidSignature if signer is not owner param
✓ permit should revert BadV for v not in {27,28}
MEDYA TOKEN - Permit (EIP-2612)
✓ creates allowance with valid permit
✓ cannot reuse same nonce (46ms)
MEDYA - rescueERC20 extra security
✓ rescueERC20 should revert CannotRescueOwnToken
✓ rescueERC20 should revert ZeroAddress when to is zero
✓ rescueERC20 should revert TokenTransferFailed when token returns false
✓ rescueERC20 should revert TokenTransferFailed when token reverts
✓ rescueERC20 should succeed and emit Rescued event (69ms)
MEDYA - Extra security regression
✓ increaseAllowance should revert on zero spender


Administrator: Windows PowerShell
MEDYA - Extra security regression
✓ increaseAllowance should revert on zero spender
✓ decreaseAllowance should revert on zero spender
✓ decreaseAllowance should revert if decreasing below zero
✓ ownership transfer to zero should revert
✓ only pendingOwner can acceptOwnership
✓ cancelRenounceOwnership requires pending
✓ startRenounceOwnership twice should revert PendingRenounceExists
✓ finalizeRenounceOwnership before ETA should revert OwnershipTimelocked
✓ after renounce, owner-only actions should revert OnlyOwner
MEDYA - Security & Negative Paths
ERC20 core negative paths
✓ transfer should revert on insufficient balance
✓ transferFrom should revert when from has insufficient balance even if allowance is enough
✓ approve should revert on zero spender
✓ transferFrom should revert on zero to
Ownership hardening
✓ transferOwnership should revert if renounce is pending
✓ startRenounceOwnership should revert if transfer is pending
✓ cancelOwnershipTransfer should revert if none pending
✓ cancelOwnershipTransfer resets ETA and pendingOwner
✓ acceptOwnership should revert when called before ETA
✓ acceptOwnership works exactly after ETA
✓ finalizeRenounceOwnership should revert when no pending renounce
✓ cancelRenounceOwnership should revert when no pending renounce
✓ cancelRenounceOwnership resets flags and eta
✓ after renounce, owner-only functions must revert for any caller
RescueERC20 safety
✓ rescueERC20 should revert when token is this contract
✓ rescueERC20 should revert when to is zero address
ETH rejection
✓ sending ETH to receive() should revert NoETH
✓ calling fallback should revert NoETH
MEDYA
✓ should have correct token metadata and initial total supply (57ms)
✓ should set the deployer as the initial owner (43ms)
✓ should transfer tokens and emit Transfer event (51ms)
✓ should approve allowance and allow transferFrom, then clear allowance (61ms)
MEDYA - view helper functions coverage
[OK] getAllowance() returned correct value.


Administrator: Windows PowerShell
MEDYA - view helper functions coverage
[OK] getAllowance() returned correct value.
✓ should return correct allowance using getAllowance()
[OK] getBalance() called successfully.
✓ should return correct balance using getBalance()
[OK] getTotalSupply() called successfully.
✓ should return total supply using getTotalSupply()
[OK] getOwner() returned correct owner.
✓ should return current owner using getOwner()
102 passing (7s)
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines
contracts\ | 100 | 92.5 | 100 | 100 |
medya.sol | 100 | 92.5 | 100 | 100 |
All files | 100 | 92.5 | 100 | 100 |
> Istanbul reports written to ./coverage/ and ./coverage.json
PS C:\xampp\htdocs\medya-token-tests> npx hardhat testnpx hardhat test