Zellic

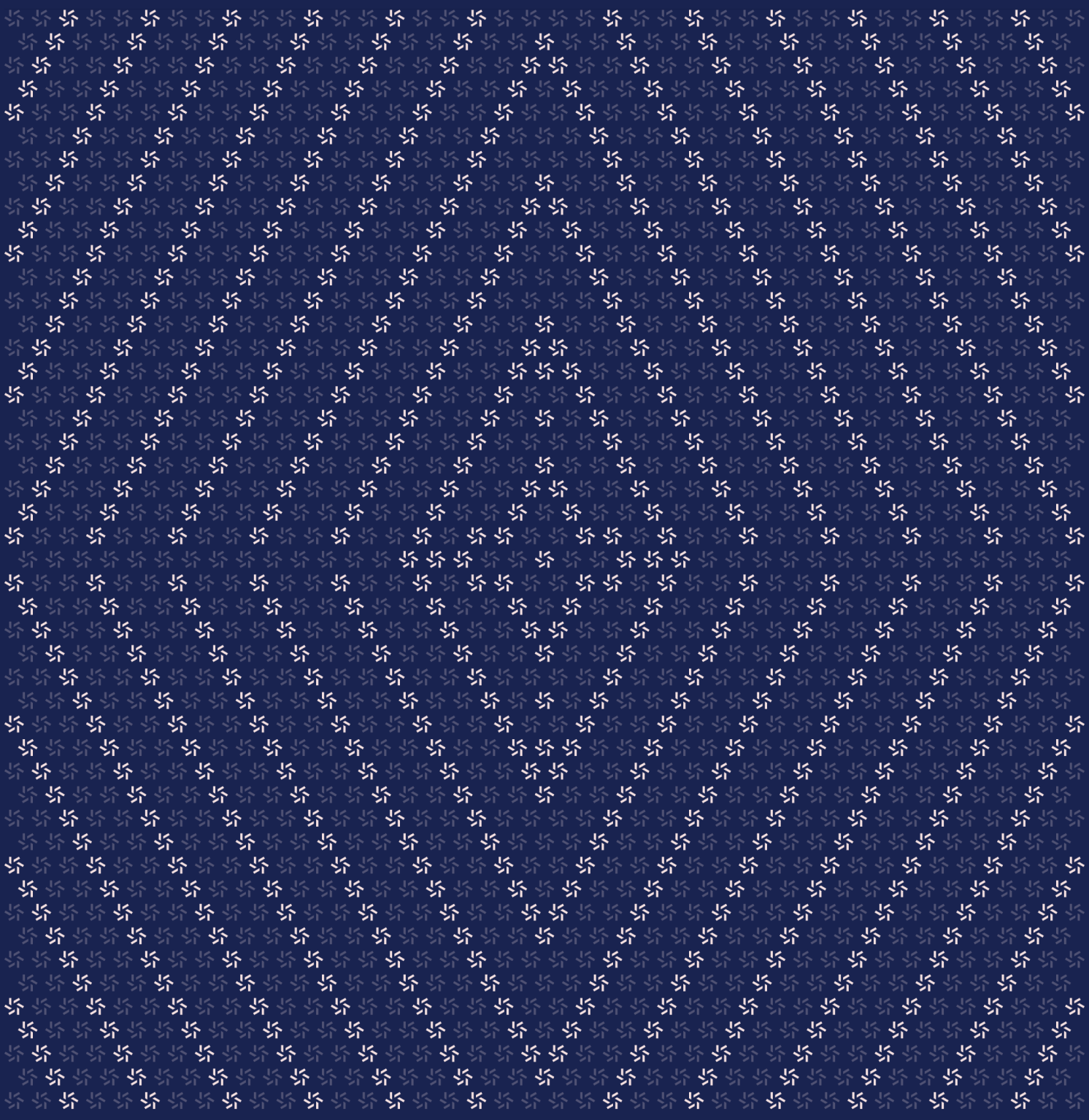
Prepared for
Brent Perreault
Jacob Torres
Paxos

Prepared by
Filipe Alves
Qingying Jie
Zellic

December 15, 2025

Paxos Token V2

Smart Contract Patch Review



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Introduction

We were asked to review the changes within commit `252d651c` ↗ of Paxos Token V2 in the enhance-signature-validation branch.

The patch adds byte array signature support for `permit`, `transferWithAuthorization`, `transferWithAuthorizationBatch`, `receiveWithAuthorization`, and `cancelAuthorization`, enabling EIP-1271 smart contract wallet signature validation by replacing `EIP712._recover` with OpenZeppelin's `SignatureChecker.isValidSignatureNow`.

It also introduces a new `cancelPermits` function, recomputes the `DOMAIN_SEPARATOR` on-the-fly to handle chain forks, and refactors shared constants, errors, and events into new `EIP2612Definitions` and `EIP3009Definitions` contracts.

1.1. Scope

The engagement involved a review of the following targets:

Paxos Token V2 Contracts

Type	Solidity
Platform	EVM-compatible
Target	paxos-token-contracts
Repository	https://github.com/paxosglobal/paxos-token-contracts ↗
Version	`cb2125103cb17d93772b0f20746eaf47707232fe`

Programs	`PaxosTokenV2.sol`
	`EIP2612.sol`
	`EIP2612Definitions.sol`
	`EIP3009.sol`
	`EIP3009Definitions.sol`
	`EIP712.sol`
	`EIP712Domain.sol`

Contact Information

The following project manager was associated with the engagement:

Chad McDonald
Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Filipe Alves
Engineer
filipe@zellic.io

Qingying Jie
Engineer
qingying@zellic.io

1.2. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.

2. Detailed Findings

2.1. Unchanged EIP-712 domain separator version permits V1 signatures on V2

Target	PaxosTokenV2		
Category	Business Logic	**Severity**	Informational
Likelihood	Low	**Impact**	Informational

Description

The `DOMAIN_SEPARATOR` function uses a hardcoded version "1" in `PaxosTokenV2`:

```
function DOMAIN_SEPARATOR() public view override returns (bytes32) {
    return EIP712._makeDomainSeparator(name(), "1");
}
```

This means unused signatures created for V1 remain valid when verified against V2. While already-used signatures are protected by nonce mechanisms (EIP-2612 uses sequential nonces, EIP-3009 marks nonces as used), any pending unused signatures from V1 would still be valid on V2.

For example, if a user signed a permit or transfer authorization on V1 but the transaction was never submitted (e.g., transaction failed, user changed their mind, or it was stored for later use), that signature could still be executed on V2 if:

1. The contract is an upgrade (same address)
2. The token name hasn't changed
3. The nonce hasn't been consumed by other operations

Impact

Unused V1 signatures remain valid in V2. The practical risk is low since it requires an upgrade scenario with pending unused signatures, but incrementing the version would provide a clean security boundary between contract versions.

This behavior may or may not be intentional. If the intent is to preserve backward compatibility with existing V1 signatures (e.g., to avoid breaking integrations that rely on pre-signed authorizations), then developers should be aware that this also means any unused V1 signatures remain valid and executable on V2. If this is not the intended behavior, the version should be incremented to invalidate all V1 signatures upon upgrade.

Recommendations

Consider incrementing the domain separator version to "2" for the new contract version. This would invalidate all V1 signatures and require users to explicitly authorize operations under the new version:

```
function DOMAIN_SEPARATOR() public view override returns (bytes32) {
    return EIP712._makeDomainSeparator(name(), "1");
    return EIP712._makeDomainSeparator(name(), "2");
}
```

Remediation

This issue has been acknowledged by Paxos.

Paxos provided the following comment:

> This is intended behavior to preserve backward compatibility with existing signatures.

3. Patch Review

The purpose of this section is to document notable and minor changes introduced in cb212510 ↗.

3.1. Notable changes

The following are the key notable changes made to the codebase.

The state variable `DOMAIN_SEPARATOR` has been replaced with a function

The state variable `DOMAIN_SEPARATOR` has been renamed to `DOMAIN_SEPARATOR_DEPRECATED` and changed to a private state variable. It is still retained for storage compatibility. The functions `_initializeDomainSeparator` and `initializeDomainSeparator`, which were used to update the old state variable, have been removed.

A new `DOMAIN_SEPARATOR` function has been added to `PaxosTokenV2`. This function recalculates the domain separator based on the current `block.chainid` each time it is queried.

```
function DOMAIN_SEPARATOR() public view override returns (bytes32) {
    return EIP712._makeDomainSeparator(name(), "1");
}
```

New function overloads

The `permit` function in `EIP2612` and the `transferWithAuthorization`, `transferWithAuthorizationBatch`, `receiveWithAuthorization`, and `cancelAuthorization` functions in `EIP3009` originally only supported signatures in (v, r, s) format. New function overloads have been added to support signatures in `bytes` format.

```
function transferWithAuthorization(
    // [...]
    uint8 v,
    bytes32 r,
    bytes32 s
) external whenNotPaused {
    // [...]
}

function transferWithAuthorization(
    // [...]
    bytes memory signature
) external whenNotPaused {
    // [...]
}
```

Update of the signature verification method

The `EIP2612` and `EIP3009` contracts previously used the `_recover` function for signature verification, which only supported signatures from EOAs. This has been updated to use `SignatureChecker.isValidSignatureNow` from OpenZeppelin, adding support for ERC-1271 and enabling signature verification when the signer is a contract. As a result, the `_recover` function and the `ECRecover` import have been removed from `EIP712.sol`.

The new function `cancelPermits`

A new `cancelPermits` function has been added to `EIP2612`. This function allows a user to invalidate a specified number of pending permits by incrementing their nonce.

```
function cancelPermits(uint256 count) external {
    if (count == 0 || count > MAX_NONCE_INCREMENT) revert InvalidNonceCount();

    _nonces[msg.sender] += count;
    emit PermitInvalidated(msg.sender, _nonces[msg.sender]);
}
```

Additionally, the `MAX_NONCE_INCREMENT` constant, `InvalidNonceCount` error, and `PermitInvalidated` event have been added to the contract `EIP2612Definitions`.

3.2. Minor changes

The following are the key minor changes made to the codebase.

Refactoring constants, errors, and events into definition contracts

Constants, errors, and events previously defined in `EIP2612` (such as `PERMIT_TYPEHASH` and `PermitExpired`) have been moved to the new `EIP2612Definitions` contract.

Similarly, constants, errors, and events previously defined in `EIP3009` (such as `TRANSFER_WITH_AUTHORIZATION_TYPEHASH`, `RECEIVE_WITH_AUTHORIZATION_TYPEHASH`, and `CallerMustBePayee`) have been moved to the new `EIP3009Definitions` contract.