

ZOC (Zombie On Chain)

ZOC: Identification and Quantification of "Zombie On Chain" Smart Contracts for Blockchain Sustainability

Abstract (Version 1.2.0)

The immutable and persistent nature of blockchains leads to an exponential accumulation of obsolete code on the ledger. Analysis indicates that up to 70% of crypto projects end up failing, leaving behind a massive volume of digital debris.

This paper proposes a new taxonomy to address this problem: the concept of the **ZOC (Zombie On Chain)** Smart Contract.

A **ZOC** is defined as a deployed contract that exhibits continuous external inactivity for an initial period of **nine months**—a threshold chosen by analogy to a complete incubation cycle—and **negligible economic value** (Criterion Y).

This delay and the value thresholds are established as an initial working hypothesis and are subject to validation and adjustment by data collected by the ZOC Tracker and future community consensus.

To guarantee robustness, we have integrated **Advanced Exclusion Criteria** to filter out false positives (such as strategically dormant Multisig and Vesting contracts). The tool capable of quantifying and classifying these entities, the **ZOC Tracker**, is architected as an **ADIP (Analytical Data Ingestion Platform)**.

1. Introduction: The Paradox of Persistence and the Need for a New Taxonomy

The promise of blockchains rests on **immutability** and **censorship resistance**. However, this permanence generates a structural consequence: the massive accumulation of **dead or abandoned code**. The problem is aggravated by the high failure rate in the ecosystem: analyses show that approximately **50% to 70% of crypto projects ultimately cease their activities**, leaving behind non-functional vestiges.

The root of the problem is structural and relies on three fundamental truths of smart contracts:

- * A poorly designed contract **cannot be corrected** after deployment.
- * It **remains immutable and public** on the ledger, sometimes even with locked funds (Locked ZOC Contracts).
- * It becomes a tangible example not to follow, but also a **valuable object** of study for research and developer education.

These structural lessons confirm the necessity of a classification. This work introduces and formally defines the concept of **Zombie On Chain (ZOC) Smart Contract**.

Figuratively, when a smart contract is deployed but serves no purpose: it is there, visible, consumes space, but is functionally lifeless.

2. Formal Definition: The ZOC Concept (Zombie On Chain)

The term **ZOC** designates smart contracts that are deemed **non-operational and economically insignificant**.

2.1. ZOC Classification Criteria

For a contract to be categorized as a **ZOC**, it must simultaneously satisfy the following two criteria:

Contract = ZOC IFF (Temporal Criterion X) \wedge (Value Criterion Y)

Temporal Criterion (Inactivity - X): Initial Period of 9 Months

The inactivity threshold is set at an initial period of **nine months (9 months)**. This period is chosen as a **working hypothesis**, based on the analogy of a complete gestation cycle, confirming abandonment or functional obsolescence.

The contract must not have registered **any external interaction** during this continuous period. On a network like Ethereum, this criterion represents approximately **2 million consecutive blocks** without activity.

Value Criterion (Economic Insignificance - Y)

The contract must present **negligible** total economic value. These thresholds are established as an **initial working basis** for the ZOC Tracker:

- Insignificant Native Balance:** The contract balance is less than 0.001 native unit of the network (e.g., 0.001 ETH).
- Minimal Secondary Assets:** The total market value of secondary assets held by the contract is **less than 10 USD**.

Methodological Note: The monetary thresholds and the nine-month period are subject to future adjustments.

The deployment of the ZOC Tracker will allow, through statistical analysis of contract populations, to optimize these criteria so that they remain relevant to the evolution of blockchain economics and transaction costs.

2.2. Classification of ZOC Types

ZOC Type	Description	Security / Exploitation Potential
 Inert	Deployed contract with no active logic, often libraries or tests.	Educational study, code design audit.
 Locked	Contract containing funds that are inaccessible due to a withdrawal bug.	Critical bug documentation, analysis of unintentional honeypots.
 Dangerous	Contract with a known, unpatched, but inactive vulnerability.	White Hat exploitation for security or exploit documentation.
 Abandoned	Contract from a dead or migrated project, but still referenced by external entities.	Traffic recovery and gentle migration via proxy contract.

3. The ZOC Tracker Architecture: The Analytical Data Ingestion Platform (ADIP)

The ZOC Tracker is more than just a dashboard; it is an **Analytical Data Ingestion Platform (ADIP)** designed to process the entirety of the EVM history with high performance and rigor. This architecture justifies the use of specialized infrastructure required to validate the **ZOC** taxonomy at Big Data scale.

3.1. The Go Ingestion Pipeline (Z-Terminal Core)

The core system is an infrastructure pipeline written in **Go (Golang)**. The choice of Go is strategic for its superior concurrency management (*goroutines*), which is essential for handling the massive importation of historical Ethereum blocks (archival catch-up) and maintaining continuous real-time monitoring.

- **Data Source:** Utilization of **Alchemy Enhanced APIs** to access the complete history and transaction traces.
- **Processing:** The Go pipeline applies the **ZOC V1.1.0 Taxonomy** logic, filtering exclusions and enriching contract metadata at the ingestion level.

3.2. Storage and Analysis: ClickHouse

The structured data is written into **ClickHouse**, an analytical, column-oriented database (OLAP).

- **Strategic Rationale:** ClickHouse is designed for massive aggregation and analytical queries on petabytes of data, making it perfect for rapidly calculating the **ZOC Score** and managing the vast dataset of **Digital Debris**.
- **Data Integrity:** This infrastructure choice ensures the separation of secrets (API keys) and guarantees the data integrity required by a **Data Delivery Manager (DDM)**.

3.2.1. Collection and Indexing of Secondary Metrics

To ensure the robustness of the analysis and to prepare for the evolution of the ZOC Score, the Go pipeline is architected to non-blockingly index several secondary metrics during the ingestion phase:

- **Complexity Measurement:** The **bytecode size** and **number of external functions** are captured to generate a **Complexity Score** to weigh the systemic audit risk of abandoned complex contracts.
- **Operational Performance:** Internal audit fields (such as **Gas Used** and **DDM Ingestion Latency**) are stored to guarantee the service's SLA (Service Level Agreement).

3.3. Exclusion Criteria (Applying the Taxonomy)

The explicit logic for filtering false positives is critical:

- This section details the analysis of **bytecode signatures** and internal contract states to identify **Strategically Dormant Contracts** (e.g., Multisig and Timelock contracts).
- This step ensures the ZOC Tracker provides an accurate metric of **true project abandonment** (and not just inactivity).

Exclusion Category	Purpose and Technical Rationale
Governance & Security Mechanisms	These contracts, such as Multisig Wallets (like Gnosis Safe) or Decentralized Autonomous Organization (DAO) Vaults , are often inactive for long periods while waiting for a quorum to be reached on a strategic proposal. If the contract's internal state (<i>storage slots</i>) indicates a pending, unexecuted transaction, the contract is flagged as Strategically Dormant (and therefore <i>not</i> a ZOC).
Time-Locked Contracts (Vesting/Timelock)	Contracts designed to hold and release assets on a predefined schedule (e.g., employee token vesting or delayed fund releases). Inactivity is an expected function until a specific future block or timestamp is reached. Exclusion is applied if the contract still holds unvested tokens or if the lockup period has not yet fully elapsed.
Infrastructure Components (Proxies/Libraries)	Contracts deployed solely as internal dependencies, such as Upgrade Proxies (e.g., UUPS or Transparent Proxy patterns) or Logic Libraries , are not meant to be called by end-users. Their primary function is to be called by other smart contracts. These must be entirely excluded from the inactivity criteria, as their lack of external calls is normal operational behavior.

Methodology for Exclusion: The identification of these exceptions is achieved through the **analysis of contract bytecode signatures** (identifying known contract patterns) and by actively **checking the contract's internal state** via Enhanced APIs (like those provided by Alchemy) to determine if a condition for activity is pending (e.g., checking if the Multisig quorum is met or if the Timelock date has expired).

This mechanism ensures that the ZOC Tracker provides a metric reflecting **true project abandonment** rather than intentional strategic quiescence.

4. Conclusion and Perspectives

The immutability of distributed ledgers is both their greatest strength and their greatest structural challenge.

This work has formalized one of the inevitable consequences of this permanence: **the silent proliferation of abandoned smart contracts**. By introducing the **ZOC** taxonomy (Zombie On Chain), we provide the first rigorous analysis framework for distinguishing persistent code from functional code.

The **ZOC Score (0-100)** synthesizes the technical and economic risk. It relies not only on Criteria X and Y (Inactivity and Value) but also integrates the **Code Complexity** and the presence of **Risk Signatures** (identified by the Security Expert) as aggravating factors.

The **ZOC Tracker** is not just a diagnostic tool; it is a step towards more mature and responsible **chain hygiene**.

By recognizing and measuring the existence of ZOCs, the technological community empowers itself to better manage, audit more accurately, and ensure greater sustainability of decentralized infrastructure.