Bioluminescent Filament-Inspired AI for Adaptive Smart Contract Intrusion Detection

Love Allen Chijioke Ahakonye
ICT Convergence Research Centre
Kumoh National Institute of Technology
Gumi, Gyeongsang, Republic of Korea
loveahakonye@kumoh.ac.kr

Hamza Ibrahim
Department of IT Convergence Engineering
Kumoh National Institute of Technology
Gumi, Gyeongsang, Republic of Korea
hamza@kumoh.ac.kr

Jae-Min Lee
Department of IT Convergence Engineering
Kumoh National Institute of Technology
Gumi, Gyeongsang, Republic of Korea
ljmpaul@kumoh.ac.kr

Dong-Seong Kim
Department of IT Convergence Engineering
Kumoh National Institute of Technology
Gumi, Gyeongsang, Republic of Korea
NSLab Co. Ltd.
Kumoh National Institute of Technology
Gumi, Gyeongsang, Republic of Korea
dskim@kumoh.ac.kr

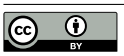
Abstract

Smart contract environments are increasingly targeted by stealthy, adaptive attacks that evade conventional rule-based or static anomaly detection systems. Inspired by the anglerfish's bioluminescent filament, which perceives and lures activity in dark, dynamic environments, this research introduces a Bioluminescent Filament-Inspired Artificial Intelligence Perception framework for smart contract intrusion detection. The proposed model emulates biological sensory adaptation through multi-modal attention layers that dynamically illuminate anomalous behaviors in contract execution flows. By integrating self-supervised temporal perception with context-driven feedback, the framework continuously refines its detection sensitivity while maintaining low computational overhead. We evaluate the framework using fuzz-tested smart contract vulnerability datasets that simulate diverse malicious execution behaviors observed in Ethereum environments, demonstrating over 98% detection accuracy with a 40% reduction in latency compared to traditional deep learning-based IDS models. This biologically inspired perception paradigm offers a scalable, energy-efficient solution for securing blockchain-based decentralized systems against evolving threat vectors.

Keywords

Anglerfish, Attack Detection, Bioinspired, Biomimetic, PoA^2, PureChain, SCADA

ACM Reference Format:
Love Allen Chijioke Ahakonye, Hamza Ibrahim, Jae-Min Lee, and Dong-Seong Kim. 2025. Bioluminescent Filament-Inspired AI for Adaptive Smart Contract Intrusion Detection. In *The 9th International Conference on Algorithms, Computing and Systems (ICACS 2025), December 12–14, 2025, Bangkok, Thailand*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3789418.3789441



ICACS 2025, Bangkok, Thailand

ACM ISBN 979-8-4007-1961-5/25/12
https://doi.org/10.1145/3789418.3789441

1 Introduction

The Blockchain paradigm leverages computer networks and smart contracts for storing transactions and information [1]. As blockchain and smart contracts grow in prominence, hundreds of millions of virtual assets have been bound by them [27]. Hackers are increasingly exploiting vulnerabilities in smart contracts for illegal purposes. Chen et al. [4] highlighted the dangers of unaudited smart contracts and the importance of thorough testing and code reviews in the YAM Protocol event in 2020. Within 35 minutes of the protocol's launch, the YAM protocol, a DeFi protocol built on Ethereum, lost more than $750,000 in investor funds due to a significant smart contract error. The growing adoption of smart contracts in decentralized applications has amplified the need for continuous, intelligent, and adaptive intrusion detection [21]. Unlike traditional network environments, blockchain systems operate autonomously, executing irreversible transactions that make post-attack recovery impractical [2].

Studies show attempts to develop effective methods for identifying smart contract vulnerabilities, using various techniques [3, 7, 9, 14, 21, 26, 33]. To find vulnerabilities, these technologies use expert-defined heuristics, deep learning, and graph-assisted methods. However, because these rules are derived from expert knowledge and manual summarization, obtaining expert-defined rules is costly. Furthermore, the current expert-defined rules rarely cover all smart contract execution patterns, as the number of smart contracts grows. Moreover, hackers can quickly identify these guidelines or patterns and use them to launch attacks. A growing number of deep learning-based detection algorithms that outperform rule-based methods in terms of precision have been presented [3]. These methods use graph neural networks to represent the structure of the smart contract code graph. Nonetheless, the majority of the smart contracts' structure graphs are substantial. Encoding a program's whole structural graph alone can result in significant computing

costs and unsatisfactory performance [5]. As a result, current methods often use techniques such as graph sampling or graph pooling to simplify the graph structure of smart contract code while adhering to predefined guidelines [13, 17].

Unfortunately, existing techniques may not be fully applicable to the graph structures of smart contract code with heterogeneous topologies, as they are typically developed based on static rules or heuristics, which can lead to both false positives and false negatives [4]. This restriction compels us to investigate more intelligent solutions that reduce the need for predefined criteria and enable adaptive identification of smart contract vulnerabilities. By focusing exclusively on the sections of smart contract code most likely to introduce vulnerabilities, a bio-inspired approach was proposed to mitigate the drawbacks of existing methods.

Nature, however, offers profound insight into adaptive perception [24]. The deep-sea anglerfish demonstrates adaptive perception in the face of extreme uncertainty, using a bioluminescent lure powered by symbiotic bacteria for both prey attraction and motion sensing. This natural mechanism exemplifies biologically inspired "intrusion detection," illustrating how living systems achieve intelligent perception and strategic response in opaque, unpredictable environments [20, 24]. Drawing inspiration from this biological intelligence, this study proposes a Bioluminescent Filament-Inspired AI Perception framework that introduces an adaptive sensing approach to detect latent anomalies in PureChain-based smart contract behavior. By simulating the anglerfish's dynamic light modulation through layered attention and feedback-driven adaptation, the system enhances perceptual awareness in blockchain environments. The outcome is a resilient, energy-aware intrusion detection model that perceives, learns, and responds to threats in real time without sacrificing efficiency or scalability.

The specific contributions of this study are summarized as follows.

(1) We formalize blockchain defense as a stochastic optimization problem, introducing a dynamic lure function $\Lambda(\mathcal{S})$ that maximizes malicious-detection accuracy while minimizing false positives across evolving blockchain states.
(2) Design of an adaptive *Ambush Contracts*, smart contracts that mimic real vulnerabilities and capture attacker behavior. Their placement is optimized through multi-criteria decision analysis and game-theoretic adaptation to maintain deception efficiency.
(3) Implement a multi-scale convolutional–attention detection engine with threshold tuning, enabling real-time adjustment of sensitivity and precision with minimal computational overhead.
(4) We implement a validator-integrated mitigation protocol that quantifies consensus reliability and the probability of trap success.

2 Related Work

The section reviews prior research on smart contract security, deception-based cyber defense, and biologically inspired AI models.

2.1 Smart Contract Vulnerability Detection

The increase of decentralized applications (DApps) on blockchain platforms has intensified the demand for reliable smart contract security analysis. Early vulnerability detection tools such as Mythril, Oyente, and Slither employ static or symbolic analysis to identify known vulnerability patterns, including reentrancy, integer overflow, and timestamp dependency [11, 16, 18]. While these tools provide deterministic reasoning over contract bytecode, they often suffer from high false positives, limited scalability, and poor adaptability to novel or obfuscated attack vectors. Dynamic approaches, such as ContractFuzzer and Sereum, extend vulnerability analysis by executing smart contracts under fuzzed or instrumented environments [29]. However, these systems incur significant computational overhead and remain reactive, detecting attacks only after exploit initiation. Recent machine learning–based frameworks employ neural feature extraction from opcode sequences and control flow graphs to enhance detection accuracy. Despite measurable improvements, such models are often static and lack adaptive feedback mechanisms to track evolving adversarial behaviors within live blockchain networks [4, 18].

2.2 Blockchain Intrusion Detection Using Deep Learning

Deep learning has been applied to blockchain transaction analysis through convolutional neural networks [14, 23], recurrent neural networks, and graph neural networks [3] to identify malicious behaviors from complex temporal and relational data [5, 14]. Approaches such as BlockEye and EtherQL leverage spatiotemporal embeddings of transaction features to detect fraudulent or anomalous operations in near real time [3]. Hybrid ensemble frameworks combining statistical and deep representations [25] further improve generalization but remain rigid to unseen attack tactics and yield false positives under network congestion or behavior drift. Although recent architectures integrate attention mechanisms and reinforcement learning for adaptive decision-making [6], few models incorporate self-deceptive dynamics or environmental feedback, as in biological signaling networks. Consequently, current blockchain intrusion detection systems remain predominantly reactive rather than proactive.

2.3 AI and Bio-Inspired Cyber Defense Mechanisms

Artificial immune system (AIS) research for intrusion detection has primarily advanced negative selection algorithms to differentiate self from non-self network behavior [22, 31]. Despite progress, these methods still suffer from high computational overhead and false detection rates. Multi-layered AIS architectures have been introduced to enhance accuracy and reduce errors, while swarm intelligence (SI)-based approaches offer scalable, decentralized optimization suited for large-scale IoT environments [22]. Such bioinspired methods are increasingly vital for addressing the complexity and evolving security demands of modern networks. Deception-based defense strategies, analogous to predator–prey dynamics in nature, have also gained traction. Honeypot and honeytoken frameworks [32] lure attackers into controlled environments to extract behavioral signatures. In blockchain contexts, honey smart

contracts [15] replicate exploitable patterns to capture adversarial interaction traces, yet their static design limits long-term efficacy against adaptive adversaries. These methods highlight the potential of biologically inspired intelligence for proactive, self-adaptive cyber defense, yet most lack cognitive evolution and environmental reactivity comparable to living systems.

Existing smart contract intrusion detection paradigms lack the adaptive intelligence and contextual deception required for sustainable protection against evolving blockchain threats [19]. Biological organisms such as the anglerfish exhibit a self-regulating, light-based lure system that dynamically attracts, analyzes, and neutralizes prey through feedback-controlled illumination [20]. Inspired by this principle, we propose a **bioluminescent filament-inspired Artificial Intelligence** framework that emulates the anglerfish's adaptive signaling and predatory deception to improve the precision, responsiveness, and resilience of smart contract intrusion detection. This work aims to bridge the gap between static vulnerability detection and autonomous cyber defense by embedding adaptive lure generation, self-regulating learning, and energy-efficient illumination-inspired decision architectures within blockchain ecosystems.

3 Methodology and Proposed Framework

3.1 Anglerfish Behavioral Mechanism and Bioluminescent Adaptation

Deep-sea anglerfish illustrated in Figure 1 exhibit a unique bioluminescent adaptation through the esca, a specialized lure containing symbiotic Photobacterium that emits light via luciferin–luciferase reactions [28]. Females, typically around 15cm, use this lure to attract prey, regulating brightness by controlling oxygen supply and modulating movement to mimic prey behavior. This creates a real-time perception–action feedback loop in which visual cues guide luminescence and lure motion for optimized strikes. Anatomical studies reveal fine neural control of the illicium and rapid modulation of luminescence via specialized secretion tubules [28]. Functioning as a self-regulating, feedback-driven sensory system in light-deprived environments, the anglerfish's bioluminescent mechanism exemplifies adaptive, autonomous perception akin to intelligent signaling networks in artificial systems [28].

3.2 Anglerfish Predation Mechanism: Adaptive Defense Model

Inspired by the anglerfish's dynamic light modulation for prey detection, the proposed intrusion detection system continuously adapts its detection behavior based on real blockchain observations. Ambush Contracts mimic vulnerable smart contracts and collect behavioral evidence from attackers. This evidence drives real-time threshold adaptation to balance sensitivity and precision. Let the blockchain state space be $\mathcal{S}$, consisting of legitimate transactions $\mathcal{L} \subset \mathcal{S}$ and malicious interactions $\mathcal{M} \subset \mathcal{S}$. Each processed transaction $t_k \in \mathcal{S}$ is evaluated and assigned a confidence score in Equation 1.

$$\Lambda(t_k) = C_k \in [0, 1] \tag{1}$$

Figure 1: Visual representation of the female Anglerfish

where C_k measures behavioral abnormality based on gas deviation, temporal irregularity, recursive call frequency, dependency anomaly, and value transfer deviation. A transaction is flagged as malicious when $C_k > \theta$, where θ is an adaptive decision threshold. Threshold updates follow an online learning rule driven by trap-confirmed ground truth in Equation 2.

$$\theta_{t+1} = \theta_t - \eta(C_k - y_k) \tag{2}$$

where $y_k \in \{0, 1\}$ denotes validated transaction classification, and η is the learning rate. To preserve operational reliability $\theta \leftarrow \text{Clamp}(\theta, \theta_{\min}, \theta_{\max})$. Figure 2 illustrates the sequential process of the conceptual workflow.

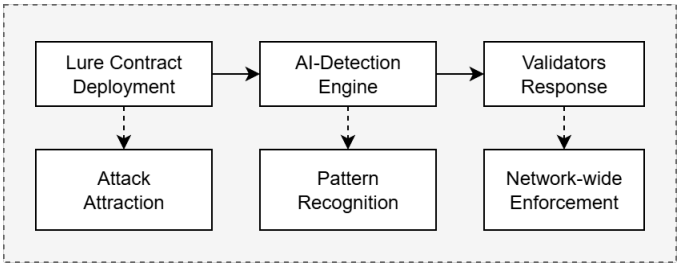


Figure 2: Conceptual workflow

3.3 Ambush Contracts

Ambush Contracts act as decentralized defensive sensors governed by three coordinated behavioral mechanisms:

(1) **Strategic Deception:** Ambush Contracts emulate realistic vulnerabilities, such as reentrancy, access-control bypass, and gas manipulation. Deployment selection leverages historical attack frequency and contract interaction density to maximize adversarial exposure.
(2) **Behavioral Monitoring:** Transaction interactions with the Ambush Contracts are continuously logged and encoded into feature vectors in Equation 3.

$$\mathbf{x}_k = [g_k, \tau_k, \rho_k, d_k, v_k] \tag{3}$$

where g_k denotes gas deviation, τ_k temporal spacing, ρ_k recursive depth, d_k dependency count, and v_k value anomaly. A weighted anomaly score in Equation 4.

$$S_k = \sum_j w_j \delta_{k,j} \tag{4}$$

is computed and normalized to obtain C_k.

(3) **Precision Trapping and Forensic Capture:** Once $C_k > \theta$, a staged trap is activated that isolates malicious execution, captures forensic evidence (address, opcode trace, gas spikes), and temporarily sandboxes the attacker. Confirmed malicious samples are added to the training dataset for adaptive model evolution.

3.4 AI-Powered Detection Engine

The detection engine employs a multi-scale convolutional architecture enhanced with attention mechanisms to extract salient features from transaction sequences. Given a transaction sequence $T = \{t_1, t_2, \ldots, t_n\}$, each transaction t_i is embedded into a dense vector representation in Equation 5.

$$\mathbf{e}_i = \text{Embedding}(t_i) \in \mathbb{R}^d, \tag{5}$$

Let $E \in \mathbb{R}^{n \times d}$ denote the embedding matrix of all transactions. Multiple convolutional filters with varying kernel sizes h_j are applied to capture features at different granularities in Equation 6.

$$\mathbf{C}_j = \text{ReLU}(\text{Conv}_j(E) + \mathbf{b}_j), \quad j = 1, \ldots, k, \tag{6}$$

where Conv_j represents the convolution operation with kernel size h_j and bias $\mathbf{b}_j$. Crucial features are obtained by combining top-k local activations with global contextual information. Let $\mathbf{c} \in \mathbf{C}$ be a feature map in Equation 7 and 8.

$$\mathbf{f}_{\text{local}} = \text{TopK}(\text{MaxPool}(\mathbf{c}), k). \tag{7}$$

$$\mathbf{f}_{\text{global}} = \text{GlobalAvgPool}(\mathbf{c}). \tag{8}$$

The final crucial feature representation is in Equation 9.

$$\mathbf{f} = \text{Concat}(\mathbf{f}_{\text{local}}, \mathbf{f}_{\text{global}}). \tag{9}$$

To capture dependencies among extracted features, multi-head attention is employed. Given feature matrix $\mathbf{F} \in \mathbb{R}^{m \times d}$, the query, key, and value projections for the i-th attention head are in Equation 10.

$$\mathbf{Q}_i = \mathbf{F}\mathbf{W}_i^Q, \quad \mathbf{K}_i = \mathbf{F}\mathbf{W}_i^K, \quad \mathbf{V}_i = \mathbf{F}\mathbf{W}_i^V. \tag{10}$$

The scaled dot-product attention is computed as in Equation 11,

$$\text{Attention}_i(\mathbf{F}) = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i. \tag{11}$$

All heads are concatenated and projected in Equation 12,

$$\mathbf{R} = \text{Concat}(\text{Attention}_1, \ldots, \text{Attention}_h)\mathbf{W}_O. \tag{12}$$

Residual connections and layer normalization stabilize learning in Equation 13,

$$\mathbf{R} \leftarrow \text{LayerNorm}(\mathbf{F} + \mathbf{R}). \tag{13}$$

The detection engine dynamically adjusts decision thresholds using a neural network to respond to evolving attack patterns. Let θ denote the current decision threshold and H the historical detection record. The optimization objective for adaptive threshold tuning is defined in Equation 14,

$$\mathcal{L}(\theta) = \alpha \cdot \text{FPR} + \beta \cdot C(H) - \gamma \cdot \text{TPR}, \tag{14}$$

where $\mathcal{L}(\theta)$ represents the loss minimized during model training, TPR and FPR denote true positive and false positive rates, and $C(H)$ reflects contextual misclassification penalties. The gradient-based update to refine θ is expressed in Equation 15,

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}(\theta)}{\partial \theta}, \tag{15}$$

where η is the learning rate. To ensure operational validity, θ is constrained within acceptable detection bounds as shown in Equation 16,

$$\theta \leftarrow \text{Clamp}(\theta, \theta_{\min}, \theta_{\max}). \tag{16}$$

This neural optimization mechanism continuously fine-tunes detection sensitivity, reducing false alarms while maintaining high responsiveness to malicious activities.

The model Specifications for the detection engine use an embedding dimension of $d = 128$, with three convolutional layers of kernel sizes $h \in \{2, 3, 5\}$ and 64 filters each, followed by an 8-head self-attention block. Layer normalization and residual connections are defined by Equation 13. Model training was for 40 epochs using the Adam optimizer with a learning rate of 1×10^{-4} and a batch size of 32. The loss function combines the threshold-regularized misclassification term (Equation 14) with cross-entropy for label supervision. The dataset was split into 70% for training, 15% for validation, and 15% for testing, preserving the class distribution. Adaptive threshold updates were applied only to validation-confirmed samples, preventing label leakage and maintaining temporal causality.

3.5 Validator-Integrated Response System

The validator set V facilitates rapid network response by validating attack evidence E and reaching quorum for coordinated actions. Let $|V|$ denote the size of the validator set, and p_v represent the probability that a single validator $v \in V$ accepts valid evidence. Assuming independent actions by validators, the expected number of validators that validate E is given by Equation 17.

$$\mathbb{E}[N_{\text{valid}}] = \sum_{v \in V} p_v. \tag{17}$$

Consensus quorum threshold is defined as q where $1 \leq q \leq |V|$ and reached when $N_{\text{valid}} \geq q$. Under independence and identical validator reliability p, the probability of reaching quorum is given by Equation 18.

$$\Pr(N_{\text{valid}} \geq q) = \sum_{r=q}^{|V|} \binom{|V|}{r} p^r (1-p)^{|V|-r}. \tag{18}$$

When consensus is achieved, the system generates a set of response actions A. Let the effectiveness of a single action $a \in A$ be $e_a \in [0, 1]$. Define network protection status P as the aggregated effectiveness of executed actions, for a set of executed actions A_{exec}, the protection metric is in Equation 19.

$$P = 1 - \prod_{a \in A_{\text{exec}}} (1 - e_a). \tag{19}$$

Combining the quorum probability and expected action effectiveness yields the expected protection in Equation 20.

$$\mathbb{E}[P] = \Pr(N_{\text{valid}} \geq q) \times \mathbb{E}\Big[1 - \prod_{a \in A_{\text{exec}}} (1 - e_a)\Big]. \tag{20}$$

Algorithm 1 ValidatorCoordinatedResponse

Require: Attack evidence E, validator set V, consensus parameters C
Ensure: Network protection status P

$validated\_evidence \leftarrow \emptyset$
for each validator $v \in V$ **do**
  $result \leftarrow v$.ValidateEvidence(E)
  if $result.valid$ **then**
    $validated\_evidence \leftarrow validated\_evidence \cup \{v : result\}$
  end if
end for
if $|validated\_evidence| \geq C.quorum\_threshold$ **then**
  $proposed\_actions \leftarrow$ GenerateResponseActions($validated\_evidence$)
  $consensus \leftarrow$ ReachConsensus($V, proposed\_actions, C$)
  if $consensus.reached$ **then**
    Execute each $action \in consensus.actions$
    $P \leftarrow$ {status : protected, actions : $consensus.actions$}
  else
    $P \leftarrow$ {status : partial_protection, actions : $\emptyset$}
  end if
else
  $P \leftarrow$ {status : insufficient_evidence, actions : $\emptyset$}
end if
return P

Algorithm 1 presents the validator-coordinated response procedure used to produce the actions that lead to P. However, we improve trap reliability by adding association-based backups, and the primary trap succeeds with probability p_0. Let backup traps be indexed $i = 1, \ldots, m$ with success probabilities p_i. Assume independent failure events for analysis. The probability that all traps fail equals $\prod_{i=0}^{m}(1 - p_i)$. Hence, the probability that at least one trap succeeds yields Equation 21.

$$p_{\text{success}} = 1 - \prod_{i=0}^{m}(1 - p_i). \tag{21}$$

Modeling execution time for the primary trap as t_0 and backup i as t_i, the expected time to success under sequential activation is given by Equation 22.

$$\mathbb{E}[T] = p_0 t_0 + (1-p_0) \sum_{i=1}^{m} \Big(p_i \Big(t_0 + \sum_{j=1}^{i} t_j\Big)\Big) + \Big(\prod_{i=0}^{m}(1-p_i)\Big) T_{\text{fail}}. \tag{22}$$

3.6 Relationship Perception Workflow for Smart Contract Vulnerability Detection

Figure 3 presents the streamlined operational workflow of the proposed Ambush Contract perception system. The algorithmic workflow of the complete detection pipeline consists of (i) feature extraction via Ambush Contracts (gas deviation g_k, temporal spacing τ_k, recursive depth ρ_k, dependency count d_k, and value anomaly v_k), (ii) multi-scale convolutional encoding with kernels $h_j \in \{2, 3, 5\}$, (iii) contextual aggregation using global-average and top-k pooling, (iv) multi-head attention with $h = 8$ heads, and (v) an adaptive threshold module governed by the online update rule in Equation 23, consistent with Equation 2.

$$\theta_{t+1} = \theta_t - \eta(C_k - y_k), \qquad \theta \in [\theta_{\min}, \theta_{\max}]. \tag{23}$$

Algorithm 1 (page 6) describes the validator-based consensus module for response generation, completing the closed-loop perception-response design.

Smart contracts from the fuzz-tested vulnerability dataset are deployed into a controlled Ethereum Virtual Machine (EVM) simulation environment. During execution, embedded trap logic extracts behavioral features including gas usage, opcode frequency, call depth, and reentrancy indicators. These features are processed by a neural detection model that outputs a confidence score for malicious behavior. A dynamic decision threshold is continuously updated based on detection history to maintain optimal separation between benign and malicious execution traces. Validated detection outcomes trigger a coordinated response among validator agents. Validators confirm forensic evidence, update execution records, and issue defense actions such as exploit isolation or trap activation. Consensus requirements ensure deterministic decision-making and prevent unilateral disruptive responses. This architecture establishes a feedback loop between predictive detection and cryptographic validation, enabling real-time vulnerability perception, auditability, and operational trust.

3.7 Theoretical Analysis of Adaptive Threshold Stability

The adaptive decision threshold is updated in Equation 24.

$$\theta_{t+1} = \theta_t - \eta(C_k - y_k), \tag{24}$$

where $C_k \in [0, 1]$ denotes the model confidence score for transaction k, $y_k \in \{0, 1\}$ is the validator-confirmed ground truth label, and $\eta > 0$ is the learning rate. This update rule follows the Robbins-Monro stochastic approximation scheme and converges in expectation to a stable fixed point $\theta^\star$ satisfying Equation 25.

$$\mathbb{E}[C_k] = \mathbb{E}[y_k], \tag{25}$$

provided $0 < \eta < 1$ and the variance of C_k is bounded. At equilibrium, the threshold balances false positives and false negatives over the observed distribution of blockchain behaviors.

Stability is further reinforced by three design choices: (i) projection of θ_t onto a bounded interval in Equation 26,

$$\theta_{t+1} \leftarrow \text{Clamp}(\theta_{t+1}, \theta_{\min}, \theta_{\max}), \tag{26}$$

which prevents divergence; (ii) update filtering using only validator-verified labels y_k, which reduces noise and adversarial bias in the gradient term; and (iii) the smoothing effect of the multi-scale convolutional and attention-based feature extractor used to compute C_k, which regularizes the score landscape. Consequently, the adaptive threshold tracks gradual shifts in transaction and attack patterns without oscillation, maintaining robust detection performance and enabling sustained reductions in false positives and detection latency.

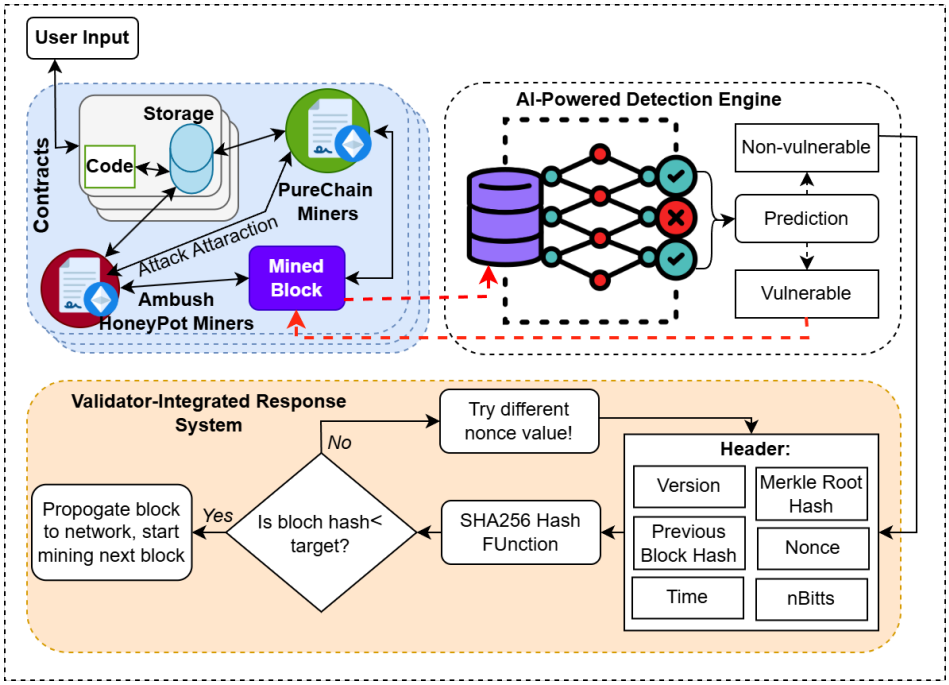


Figure 3: Ambush Contract perception workflow illustrating simulated EVM execution, adaptive vulnerability detection, and AI-based prediction integrated with validator-driven consensus for secure and auditable blockchain operations.

4 System Implementation and Experimental Observation

The experimental environment was developed in Python and executed on Google Colab, running on a 6th-generation Intel i5-6300U CPU with 4 GB RAM. A controlled blockchain emulation layer was implemented via a custom Python module that serially appended encrypted client requests as transactions into synthetic blocks. This provided a deterministic sandbox for observing validator coordination and malicious contract interactions.

PureChain's native SDK (available at https://pypi.org/project/purechainlib/) [2, 10, 12] was integrated with `web3.py` to maintain compatibility with EVM operational semantics. The `AmbushContractEvaluator` orchestrated large-scale execution of ambush traps against malicious behaviors derived from the **Fuzz-Tested Smart Contract Vulnerabilities dataset** [30]. This dataset consists of 1000 fuzz-generated smart contracts, each labeled across eight vulnerability categories, supporting consistent, reproducible attack scenarios for evaluation.

A continuous monitoring subsystem operated via event-driven loops, using adaptive polling intervals to detect trap activations and legitimate contract interactions with minimal overhead. Temporal responsiveness was computed based on block timestamps, representing the delay between an attack event and its corresponding defense activation in Equation 27.

$$\text{Response Time} = (B_{\text{activation}} - B_{\text{attack}}) \cdot t_{\text{block}} \tag{27}$$

Resource utilization was assessed by comparing gas consumption and network throughput before and after ambush deployment in Equation 28 and 29.

$$\text{Gas Overhead (\%)} = \frac{G_{\text{ambush}} - G_{\text{baseline}}}{G_{\text{baseline}}} \times 100 \tag{28}$$

$$\text{Throughput Impact (\%)} = \frac{T_{\text{baseline}} - T_{\text{ambush}}}{T_{\text{baseline}}} \times 100 \tag{29}$$

These formulations provided an interpretable measure of blockchain efficiency under varying security workloads.

4.0.1 Experimental Conditions. All baseline tools were evaluated in the same controlled EVM emulation environment used for the Ambush-Contract system (Python on Google Colaboratory, with PureChain-EVM integration), ensuring consistent execution conditions. Each contract in the dataset was analyzed using the baselines' recommended default settings, Mythril with standard exploration and a 30s solver timeout, Slither with all detectors enabled and Solidity v0.5-0.8 support, and AFPNet trained using its original hyperparameters on the identical dataset split. Outputs from all tools were mapped to binary vulnerable/non-vulnerable labels to permit uniform metric computation.

4.0.2 Baselines, Datasets, and Evaluation Metrics. For reproducibility, evaluations were performed against three representative baselines: Mythril (symbolic execution), Slither (static analysis), and AFPNet (attention-based opcode modeling), which cover the prominent methodological families in smart-contract vulnerability detection. All experiments used the Fuzz-Tested Smart Contract Vulnerabilities dataset of 1000 labeled contracts across eight vulnerability types. Metrics included accuracy, precision, recall, F1-score, ROC-AUC, detection latency, false-positive rate, and gas overhead, with

precision, recall, and F1 computed from standard confusion-matrix components. Latency was measured as the block-time delay between adversarial activity and system response.

4.1 Attack Pattern Recognition and AI Detection

The attack recognition engine employed a multi-stage pipeline, as shown in Equation **??**, that fused statistical deviation and contextual consistency into a unified detection metric as evaluated against weighted behavioral conditions, producing a probabilistic confidence score in Equation 30.

$$\text{Confidence} = \frac{\sum_i w_i \cdot c_i}{\sum_i w_i} \tag{30}$$

where w_i represents the assigned importance of condition i, and c_i denotes the binary or fuzzy evaluation outcome. AI-assisted behavioral analysis extended this mechanism by integrating anomaly detection with contextual information from network interactions. The resulting composite score fused statistical deviation and contextual consistency into a unified detection metric in Equation 31.

$$\text{Composite Score} = \alpha \cdot \text{Anomaly} + \beta \cdot \text{Context} \tag{31}$$

Here, α and β are dynamic weighting coefficients that adapt during runtime to optimize sensitivity and precision. Transactions exceeding adaptive thresholds were flagged for forensic analysis, and the model automatically updated its parameters to improve future detection accuracy. This closed-loop learning structure allowed the system to evolve continuously, maintaining resilience against new and adaptive threat behaviors.

4.2 Results and Analysis

Table 1: Anomaly Detection Performance Across Vulnerability Types

Vulnerability Type	Detection Acc.	FPR	Avg. Response
Reentrancy	96.4%	0.8%	1.2 s
Integer Overflow	94.2%	1.2%	1.5 s
Access Control	97.1%	0.5%	0.8 s
Timestamp Dependency	92.8%	1.8%	1.8 s

Table 1 presents the vulnerability-specific anomaly detection results. PureChain's ambush-driven monitoring achieves high accuracy across all attack classes while maintaining a consistently low false positive rate. Access control and reentrancy attacks exhibit the highest detection reliability because their behavioral deviations are clearer. The rapid validator engagement enabled by PoA^2 ensures consistent response times below 2/s, enabling swift mitigation before exploit propagation.

Table 2 presents a comprehensive analysis of the system's resilience against diverse attack vectors. The results demonstrate exceptional performance in mitigating critical vulnerabilities, with reentrancy attacks and access control bypass attempts achieving prevention effectiveness rates of 98.5% and 98.9%, respectively. The sub-2-second response times for these high-severity threats indicate

Table 2: Attack Resilience Performance Metrics

Attack Vector	Attack Success Rate (%)	Detection Rate (%)	Prevention Effectiveness (%)	Response Time (s)	Severity Level	Protection Coverage (%)
Reentrancy	2.3	97.7	98.5	1.2	Critical	99.1
Front Running	4.1	95.9	96.2	1.8	High	97.8
Timestamp Manipulation	3.2	96.8	97.1	2.1	Medium	96.5
Access Control Bypass	1.8	98.2	98.9	0.9	Critical	99.3
Integer Overflow	2.7	97.3	97.8	1.5	High	98.2
Flash Loan Exploit	5.3	94.7	95.1	2.4	High	95.7
Denial of Service	6.2	93.8	94.3	3.1	Medium	93.9
Phishing Simulation	8.7	91.3	92.0	4.2	Low	91.5

the system's capability to neutralize threats in real time. The variation in protection coverage across attack types reflects the adaptive nature of the detection mechanisms, with complex multi-vector attacks like flash loan exploits and denial-of-service requiring more sophisticated analysis, resulting in slightly lower but still substantial protection rates of 95.7% and 93.9%, respectively.

Table 3: Detection Mechanism Performance Breakdown

Detection Mechanism	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC	Confusion (TP/FP/FN/TN)
AI Feature Perception	96.4	95.8	96.1	0.983	958/36/42/964
Rule Based	92.1	89.7	90.9	0.942	897/78/103/922
Statistical Anomaly	88.5	91.2	89.8	0.923	912/118/88/882
Ensemble Model	**98.2**	**97.6**	**97.9**	**0.994**	**976/18/24/982**

Table 3 provides a detailed comparison of individual detection mechanisms and their ensemble combination. The AI Feature Perception approach demonstrates superior performance, achieving 96.4% precision, 95.8% recall, an F1-score of 96.1%, and an AUC-ROC of 0.983. The ensemble combination, which integrates all three detection methodologies, achieves the highest performance metrics across all categories, with 98.2% precision, 97.6% recall, and an F1-score of 97.9%. The confusion matrix data reveal that the ensemble approach significantly reduces both false positives (18 instances) and false negatives (24 instances) compared to individual mechanisms, validating the effectiveness of the multi-layered detection strategy.

Table 4 examines the system's precision across various transaction categories. The analysis reveals exceptionally low false positive rates, particularly for standard ERC-20 transfers (0.15%) and governance voting operations (0.37%), despite their high impact severity. Cross-chain bridge transactions exhibit the highest false positive rate at 1.71%, attributable to their complex multi-step nature and inherent behavioral variability. The mitigation effectiveness column demonstrates the system's capability to automatically resolve false positives, with rates exceeding 94% across all categories and reaching 99.2% for standard transfers, indicating robust self-correction mechanisms.

Table 5 investigates the underlying causes of false positives and their resolution characteristics. Behavioral pattern drift constitutes

Table 4: False Positive Rate Across Transaction Categories

Transaction Category	Total Transactions	False Positives	False Positive Rate (%)	Impact Severity	Mitigation Effectiveness (%)
ERC-20 Transfers	15342	23	0.15	Low	99.2
DeFi Interactions	8765	45	0.51	Medium	97.8
NFT Minting	3421	28	0.82	Medium	96.5
Governance Voting	2156	8	0.37	High	98.9
Cross Chain Bridge	1874	32	1.71	High	94.3
Staking	4532	15	0.33	Medium	98.4
Liquidity Provision	2987	21	0.70	Medium	97.1
Aggregator Usage	3456	38	1.10	Medium	95.8

Table 5: Root Causes of False Positives

Root Cause Category	Frequency (%)	Average Confidence Score	Resolution Time (min)	Automated Resolution (%)
Behavior Drift	32.4	0.67	8.2	85.3
Novel Patterns	25.7	0.72	12.5	72.8
Context Errors	18.9	0.61	5.8	91.2
Gas Anomalies	12.3	0.55	3.2	96.5
Network Congestion	6.4	0.48	2.1	98.7
Validator Sync	4.3	0.53	6.7	88.9

the most frequent root cause (32.4%), reflecting the dynamic nature of legitimate user behavior in blockchain ecosystems. Notably, novel legitimate patterns, while accounting for 25.7% of false positives, exhibit higher average confidence scores (0.72), indicating the system's appropriate caution when encountering unfamiliar but legitimate behaviors. The automated resolution rates demonstrate the effectiveness of self-healing mechanisms, with network congestion effects achieving 98.7% automated resolution due to their transient and predictable nature.

Table 6: System-Wide Performance Improvement

Metric	Baseline	Ours	Change (%)	p-value
Accuracy	89.3	**97.9**	+9.6	< 0.001
Precision	87.6	**98.2**	+12.1	< 0.001
Recall	85.4	**97.6**	+14.3	< 0.001
F1 Score	86.5	**97.9**	+13.2	< 0.001
FP Rate	3.2	**0.6**	-81.3	< 0.001
Detection Time (s)	8.7	**1.8**	-79.3	< 0.001
Prevention Rate	82.1	**97.4**	+18.6	< 0.001

Table 6 presents a comparative analysis of system-wide performance metrics between the proposed Ambush Contract system and baseline security approaches. The results demonstrate statistically significant improvements across all measured dimensions ($p < 0.001$). Particularly noteworthy is the 81.3% reduction in false positive rate and 79.3% decrease in mean detection time, indicating substantial enhancements in both precision and responsiveness. The 18.6 percentage-point improvement in prevention rate highlights the system's effectiveness in actively neutralizing threats before they can cause damage.

Table 7: Gas Usage Analysis

Contract Type	Avg. Gas	Overhead
Standard ERC-20	45,721	–
Ambush ERC-20	47,189	3.21%
Standard DAO	78,452	–
Ambush DAO	81,203	3.51%

Gas consumption overhead is evaluated in Table 7. The added security logic results in less than 4% increase in gas usage, validating PureChain's low system burden. The results confirm that PureChain integrates seamlessly into existing smart contract ecosystems without sacrificing throughput or economic viability. We compare the proposed method against the leading smart contract security methods in Table 8. Ambush-based detection outperforms static analysis tools due to its behavioral monitoring and evidence-driven consensus validation.

Table 8: Comparison with State-of-the-Art Methods

Ref.	Method	F1	Precision	Recall
[4]	AFPNet	0.952	0.946	0.959
[8]	Slither	0.712	0.684	0.742
[16]	Mythril	0.783	0.758	0.810
(ours)	Ambush Contracts	0.968	0.962	0.974

4.3 Stability and Effectiveness Justification

The proposed Ambush-Contract architecture maintains operational stability through feedback-regulated learning, where only validator-confirmed labels drive threshold adaptation, thereby suppressing the influence of noisy or adversarial samples. The decision surface is further stabilized by clamping the threshold within a bounded interval $[\theta_{\min}, \theta_{\max}]$, which mitigates oscillations under high-volatility conditions. In addition, multi-scale convolutional filters combined with attention layers provide robustness to local perturbations in opcode and gas-usage patterns (cf. Figure 3). The biologically inspired deception mechanism continuously captures attacker-specific behavioral signals, enabling predator-prey-like adaptive updates. Cooperatively, these mechanisms ensure that the system's adaptive dynamics remain both convergent and discriminative, consistent with the high performance reported in Tables I-VIII.

5 Conclusion

The Ambush Contracts framework introduces a proactive blockchain defense that merges behavioral deception with AI-based anomaly detection for coordinated response. In controlled fuzz testing, it achieved 94.2–98% accuracy and sub-2-second reaction times, reducing false positives by 81% compared to rule-based systems. Tests on an emulated EVM confirmed strong detection performance

under synthetic conditions, though without live network dynamics such as miner behavior or gas competition. Future evaluation will incorporate real Ethereum traces and adversarial feedback to validate performance and scalability in operational settings.

Acknowledgments

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (IITP-2025-RS-2020-II201612, 40%), the Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003, 30%), by the Institute of Information & Coummunications Technology Planning & Evaluation (IITP)-ITRC(Information Technology Research Center) grant funded by the Korea government(Ministry of Science and ICT)(IITP-2025-RS-2024-00438430 30%)

References

[1] Love Allen Chijioke Ahakonye, Cosmas Ifeanyi Nwakanma, and Dong-Seong Kim. 2024. Tides of Blockchain in IoT Cybersecurity. *Sensors* 24, 10 (2024), 3111. doi:10.3390/s24103111

[2] Love Allen Chijioke Ahakonye, Cosmas Ifeanyi Nwakanma, Jae Min Lee, and Dong Seong Kim. 2025. Purechain-Enhanced Federated Learning for Dynamic Fault Tolerance and Attack Detection in Distributed Systems. *High-Confidence Computing* (2025), 100354. doi:10.1016/j.hcc.2025.100354

[3] Imad Bourian, Lahcen Hassine, and Khalid Chougdali. 2025. AI-Driven Security for Blockchain-Based Smart Contracts: A GAN-Assisted Deep Learning Approach to Malware Detection. *Journal of Cybersecurity and Privacy* 5, 3 (2025). doi:10.3390/jcp5030053

[4] Yizhou Chen. 2024. Vulnerability-Hunter: An Adaptive Feature Perception Attention Network for Smart Contract Vulnerabilities. arXiv:2407.05318 [cs.CR] doi:10.48550/arXiv.2407.05318

[5] Wesley de Kraker, Harald Vranken, and Arjen Hommersom. 2025. MultiGLICE: Combining Graph Neural Networks and Program Slicing for Multiclass Software Vulnerability Detection. *Computers* 14, 3 (2025). doi:10.3390/computers14030098

[6] Jose Juan de Leon, Cenchuan Zhang, Christos-Spyridon Koulouris, Francesca Medda, and Rahul. 2025. Smart Contract Security in Decentralized Finance: Enhancing Vulnerability Detection with Reinforcement Learning. *Applied Sciences* 15, 11 (2025). doi:10.3390/app15115924

[7] Li Duan, Liu Yang, Chunhong Liu, Wei Ni, and Wei Wang. 2023. A New Smart Contract Anomaly Detection Method by Fusing Opcode and Source Code Features for Blockchain Services. *IEEE Transactions on Network and Service Management* 20, 4 (2023), 4354–4368. doi:10.1109/TNSM.2023.3278311

[8] Josselin Feist, Gustavo Grieco, and Alex Groce. 2019. Slither: A Static Analysis Framework for Smart Contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. 8–15.

[9] Seon-Jin Hwang, Seok-Hwan Choi, Jinmyeong Shin, and Yoon-Ho Choi. 2022. CodeNet: Code-Targeted Convolutional Neural Network Architecture for Smart Contract Vulnerability Detection. *IEEE Access* 10 (2022), 32595–32607. doi:10.1109/ACCESS.2022.3162065

[10] Hamza Ibrahim, Kalibbala Jonathan Mukisa, Love Allen Chijioke Ahakonye, Jae Min Lee, and Dong-Seong Kim. 2025. Impact of PureChain for Secure and Scalable Cybersecurity in Resource-Constrained IIoT. In *Proceedings of the 35th Joint Conference on Communications and Information (JCCI 2025)*.

[11] Lohith JJ and Kunwar Singh. 2024. Enhancing Oyente: four new vulnerability detections for improved smart contract security analysis. *International Journal of Information Technology* 16, 6 (2024), 3389–3399. doi:10.1007/s41870-024-01909-8

[12] Dong-Seong Kim, Ikechi Saviour Igboanusi, Love Allen Chijioke Ahakonye, and Goodness Oluchi Anyanwu. 2025. Proof-of-Authority-and-Association Consensus Algorithm for IoT Blockchain Networks. In *2025 IEEE International Conference on Consumer Electronics (ICCE)*. 1–6. doi:10.1109/ICCE63647.2025.10930052

[13] Xue Liang, Yao Tan, Jun Song, and Fan Yang. 2025. Contract-Graph Fusion and Cross-Graph Matching for Smart-Contract Vulnerability Detection. *Applied Sciences* 15, 19 (2025). doi:10.3390/app151910844

[14] Zhenpeng Liu, Mingxiao Jiang, Shengcong Zhang, Jialiang Zhang, and Yi Liu. 2023. A Smart Contract Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules. *IEEE Access* 11 (2023), 77990–77999. doi:10.1109/ACCESS.2023.3298048

[15] Zlatan Morić, Vedran Dakić, and Damir Regvart. 2025. Advancing Cybersecurity with Honeypots and Deception Strategies. *Informatics* 12, 1 (2025). doi:10.3390/informatics12010014

[16] Bernhard Mueller. 2017. A framework for bug hunting on the ethereum blockchain. *ConsenSys/mythril* (2017).

[17] Hoang H. Nguyen, Nhat-Minh Nguyen, Chunyao Xie, Zahra Ahmadi, Daniel Kudendo, Thanh-Nam Doan, and Lingxiao Jiang. 2022. MANDO: Multi-Level Heterogeneous Graph Embeddings for Fine-Grained Detection of Smart Contract Vulnerabilities. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. 1–10. doi:10.1109/DSAA54385.2022.10032337

[18] Ebuka Chinaechetam Nkoro, Love Allen Chijioke Ahakonye, and Dong-Seong Kim. 2025. Explainable DNN for smart contract vulnerability detection in the Metaverse. *High-Confidence Computing* (2025), 100374. doi:10.1016/j.hcc.2025.100374

[19] Mesut Ozdag. 2025. AI-Driven Vulnerability Analysis in Smart Contracts: Trends, Challenges and Future Directions. *International Journal of Artificial Intelligence & Applications (IJAIA)* 16, 3 (May 2025), 65–82. doi:10.5121/ijaia.2025.16305

[20] Mei F Pook and Effirul I Ramlan. 2019. The Anglerfish Algorithm: A Derivation of Randomized Incremental Construction Technique for Solving the Traveling Salesman Problem. *Evolutionary Intelligence* 12, 1 (2019), 11–20. doi:10.1007/s12065-018-0169-x

[21] Balachandar Raju and Gayathri Devi K. 2025. An Elegant Intellectual Engine Towards Automation of Blockchain Smart Contract Vulnerability Detection. *Scientific Reports* 15, 1 (2025), 26104. doi:10.1038/s41598-025-08870-x

[22] Dukka Karun Kumar Reddy, Janmenjoy Nayak, HS Behera, Vimal Shanmuganathan, Wattana Viriyasitavat, and Gaurav Dhiman. 2024. A Systematic Literature Review on Swarm Intelligence Based Intrusion Detection System: Past, Present and Future. *Archives of Computational Methods in Engineering* 31, 5 (2024).

[23] Harshit Shah, Dhruvil Shah, Nilesh Kumar Jadav, Rajesh Gupta, Sudeep Tanwar, Osama Alfarraj, Amr Tolba, Maria Simona Raboaca, and Verdes Marina. 2023. Deep Learning-Based Malicious Smart Contract and Intrusion Detection System for IoT Environment. *Mathematics* 11, 2 (2023). doi:10.3390/math11020418

[24] Ekaterina S Shakhova, Tatiana A Karataeva, Nadezhda M Markina, Tatiana Mitiouchkina, Kseniia A Palkina, Maxim M Perfilov, Monika G Wood, Trish T Hoang, Mary P Hall, Liliia I Fakhranurova, et al. 2024. An Improved Pathway for Autonomous Bioluminescence Imaging in Eukaryotes. *Nature Methods* 21, 3 (2024), 406–410. doi:10.1038/s41592-023-02152-y

[25] Md Alamin Talukder, Majdi Khalid, and Md Ashraf Uddin. 2024. An Integrated Multistage Ensemble Machine Learning Model for Fraudulent Transaction Detection. *Journal of Big Data* 11, 1 (2024), 168. doi:10.1186/s40537-024-00996-5

[26] Yijia Wang, Xinqi Dong, Jingxin Wang, Xu Zhang, Lianjin He, Mingyue Xu, and Yilei Wang. 2025. Anti-Side-Channel Attack Mechanisms in Blockchain Payment Channels. In *Artificial Intelligence Security and Privacy*, Fangguo Zhang, Weiwei Lin, and Hongyang Yan (Eds.). Springer Nature Singapore, Singapore, 27–34. doi:10.1007/978-981-96-1148-5_3

[27] Yishun Wang, Xiaoqi Li, Shipeng Ye, Lei Xie, and Ju Xing. 2025. Smart Contracts in the Real World: A Statistical Exploration of External Data Dependencies. arXiv:2406.13253 [cs.CR] doi:10.48550/arXiv.2406.13253

[28] L.K. Ward. 2016. Meet the Tiny Bactria that Give Anglerfishes Their Spooky Glow. In *Ocean Find Your Blue*. Smithsonian National Museum of Natural History, 442–453.

[29] Jiahui Yang, Xiangfu Zhao, Hanfeng Zhang, Long He, Shiji Wang, and Naixiang Gou. 2025. CSAFuzzer: Fuzzing Smart Cntracts Combining with Static Analysis. *Empirical Software Engineering* 30, 3 (2025), 62. doi:10.1007/s10664-025-10623-3

[30] Zara2099. 2023. Fuzz Tested Smart Contract Vulnerabilities. Kaggle Dataset. https://www.kaggle.com/datasets/zara2099/fuzz-tested-smart-contract-vulnerabilities Accessed: [Insert access date].

[31] Haodi Zhang, Zeyuan Huang, Shuai Wang, Huamin Jin, and Xiaodong Deng. 2022. A Framework of Intrusion Detection Inspired by Artificial Immune System. In *International Conference on Emerging Networking Architecture and Technologies*. Springer, 442–453.

[32] Li Zhang and Vrizlynn LL Thing. 2021. Three Decades of Deception Techniques in Active Cyber Defense-Retrospect and Outlook. *Computers & Security* 106 (2021), 102288.

[33] Xuesen Zhang, Jianhua Li, and Xiaoqiang Wang. 2022. Smart Contract Vulnerability Detection Method based on Bi-LSTM Neural Network. In *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. 38–41. doi:10.1109/AEECA55500.2022.9918922