

Pre-Authorization Screening of EIP-7702 Account Delegation in Ethereum Blockchain under an Execution-Audited Adaptive Adversary

M M Imran¹, Anik Tahabilder¹, Changxin Bai², Qiang Zhu³, Shiyong Lu¹

¹Wayne State University, Detroit, MI, USA: {mmimran, tahabilderanik, shiyong}@wayne.edu

²Kettering University, Flint, MI, USA: cbail@kettering.edu

³University of Michigan-Dearborn, Dearborn, MI, USA: qzhu@umich.edu

Abstract—EIP-7702 allows an externally owned Ethereum account to delegate execution to smart-contract code, creating a new security decision before authorization. At this stage, a delegate may have little transaction history, reputation, or verified source code, while its runtime bytecode is already available for screening. Existing learning-based contract analysis is commonly evaluated on clean inputs or predefined transformations, which does not capture a bytecode author who can adapt the program in response to a screener. We study pre-authorization screening under this adaptive threat and present AuthGuard-7702, a bytecode-only framework built around AuthGuard-Seq, a hierarchical opcode model with selective chunk aggregation. We evaluate AuthGuard-7702 on 2,190 EIP-7702 delegates using family-disjoint splits and a score-aware adversary that searches over compositions of bytecode transformations. The primary attack space is supported by a real-EVM preservation audit covering 5,196 transformed executions from 120 bytecode families. Clean AUPRC does not statistically separate AuthGuard-Seq from a 15-feature logistic-regression screener, but execution-audited adaptive evaluation reveals a substantial robustness difference. Under the primary adversary, paired attack success is 0.167 for AuthGuard-Seq compared with 0.537 for LR, while robust recall is 0.679 versus 0.378. Parameter-matched adaptive controls further associate this robustness with selective chunk aggregation rather than model capacity alone. Overall, AuthGuard-7702 shows that preservation-aware adaptive evaluation can reveal security-relevant model differences that clean evaluation misses at the EIP-7702 authorization boundary.

Index Terms—EIP-7702, smart contract security, bytecode analysis, adversarial robustness, pre-authorization screening

I. INTRODUCTION

Ethereum traditionally separates externally owned accounts (EOAs), controlled by user keys, from smart contracts containing executable logic. EIP-7702 allows an EOA to delegate execution to smart-contract code while retaining its address [1]. This enables transaction batching and gas sponsorship but creates a new security decision before authorization. Delegate code may gain substantial authority over assets, approvals, and future transactions while conventional trust signals remain limited. A newly deployed delegate may have little transaction history, no established reputation, and no verified source code, yet its runtime bytecode is already available on chain. A wallet or security service can therefore screen the proposed delegate before the user signs the authorization (Figure 1).

In pre-authorization screening, the party seeking to evade the screener controls the inspected program. A delegate

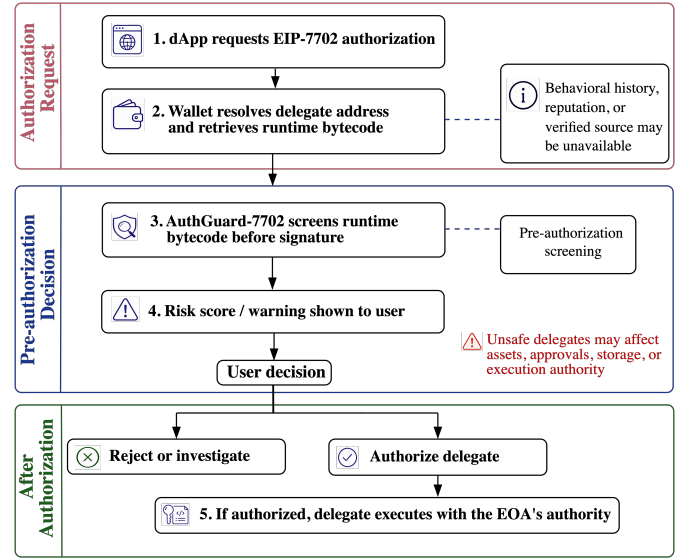


Fig. 1. EIP-7702 pre-authorization screening window

author can modify bytecode, redeploy variants, and probe a screening service before presenting a candidate for authorization. Performance on held-out clean contracts may therefore change substantially under deliberate adaptation. Adversarial robustness is consequently part of model selection rather than a secondary stress test.

Prior work has shown that contract bytecode can support vulnerability, phishing, and malicious-contract detection [2]–[5], while more recent studies examine robustness to obfuscation and adversarial perturbations [6]–[8]. These studies establish the relevance of both bytecode learning and adversarial robustness, but they do not evaluate score-aware adaptive pre-authorization screening at the EIP-7702 authorization boundary. This setting also introduces a second requirement. An apparent evasion is security-relevant only when the transformed bytecode preserves the behavior that motivated the original screening decision. Without preservation evidence, a successful score reduction may reflect a broken program rather than a meaningful bypass. We therefore ask: *how should a bytecode screener for EIP-7702 delegates be evaluated when the delegate author can actively adapt the program to evade it?* We address this question with AuthGuard-7702, a bytecode-only pre-authorization screening

framework built around AuthGuard-Seq, a hierarchical opcode model with selective chunk aggregation. AuthGuard-7702 evaluates a score-aware attacker that searches over compositions of bytecode transformations under bounded query and size constraints. The primary adaptive attack space is grounded in an independent real-EVM preservation audit. Across 5,196 transformed executions from 120 bytecode families, flooding, metadata rewriting, and neutral instruction insertion preserve the measured execution behavior. These audit-supported transformations define the primary adversary, while transformations with weaker preservation evidence are evaluated separately as a broader structural stress test.

The resulting evaluation exposes a model-selection difference that clean performance alone does not reveal. On a family-disjoint benchmark of 2,190 EIP-7702 delegates, AuthGuard-Seq and a 15-feature logistic-regression screener are not statistically separated by clean AUPRC. Under the primary execution-audited adaptive adversary, however, paired attack success is 0.167 for AuthGuard-Seq compared with 0.537 for LR, while robust recall is 0.679 versus 0.378. The paired robustness advantage remains positive across MinHash family thresholds of 0.80, 0.85, and 0.90. Parameter-matched adaptive controls and attention-allocation analysis further identify selective chunk aggregation as the strongest architectural explanation for this robustness within the evaluated model family.

This study makes three contributions:

- We formulate EIP-7702 pre-authorization screening under a score-aware, query-bounded adversary and ground the primary attack space in a real-EVM preservation audit covering 120 bytecode families and 5,196 transformed executions.
- We show that clean AUPRC does not statistically separate AuthGuard-Seq from the 15-feature LR screener, while adaptive evaluation yields a paired ASR advantage of 0.370 that remains positive across alternative bytecode-family definitions.
- We use parameter-matched adaptive controls and attention-allocation analysis to associate the robustness advantage with selective chunk aggregation rather than model capacity or full-stream coverage alone.

II. BACKGROUND AND RELATED WORK

This section reviews the EIP-7702 authorization boundary, learning-based smart-contract analysis, adversarial robustness, and functionality-preserving evasion.

A. EIP-7702 and the Authorization Boundary

EIP-7702 allows an externally owned account (EOA) to delegate execution to code deployed at another address while retaining its original account address [1]. A valid authorization associates the EOA with a delegation indicator of the form `0xef0100 || address`, after which execution resolves the referenced runtime code in the EOA’s account context. Because the delegate address and its runtime bytecode are available before the authorization is signed, delegate bytecode provides a natural basis for pre-authorization screening. Recent work

TABLE I
POSITIONING AGAINST REPRESENTATIVE PRIOR WORK

Work	EIP-7702	Bytecode only	Pre-auth. screening	Attack model	Preservation audited
Huang et al. [9]	✓	–	–	–	–
Qi et al. [10]	✓	–	–	–	–
PhishingHook [4]	–	✓	–	–	–
Eth2Vec [3]	–	✓	–	–	–
ContractWard [2]	–	✓	–	–	–
FinDet [6]	–	✓	–	fixed	–
ContractShield [7]	–	–	–	fixed	–
ORACAL [8]	–	–	–	hard-label	–
AuthGuard-7702	✓	✓	✓	score-aware	✓

“Fixed” denotes predefined perturbations or obfuscations. “Hard-label” denotes black-box attacks that observe only model decisions. “Score-aware” denotes an attacker that observes screening scores and searches over combinations of bytecode transformations. “Preservation audited” denotes execution-level validation of transformations supporting the primary robustness claim.

demonstrates the security importance of this authorization boundary. Huang et al. [9] combine transaction analysis and cross-contract static analysis to characterize security risks associated with EIP-7702 delegation, while Qi et al. [10] study delegation-based phishing attacks and corresponding defenses. These studies establish that delegate behavior requires careful security analysis. AuthGuard-7702 addresses a complementary question: whether runtime bytecode can be screened before authorization and how such a screener behaves when the delegate author actively adapts the bytecode to evade detection.

B. Learning-Based Analysis and Adversarial Robustness

Learning-based methods have been widely explored for smart-contract security analysis using different representations and supervision signals. ContractWard uses opcode bigrams for vulnerability detection [2], Eth2Vec learns distributed representations directly from EVM bytecode [3], and PhishingHook studies opcode-based representations for phishing detection [4]. DLVA predicts vulnerability labels directly from EVM bytecode using source-level Slither analysis as supervision [5], while SMARTCAT identifies price-manipulation attack contracts from bytecode without requiring source code or transaction data [11]. These systems demonstrate the value of bytecode-level learning, but their primary objective is predictive detection rather than pre-authorization robustness against an adaptive bytecode author. Evaluation methodology is particularly important for learned program analysis. Prior work shows that representation choices, dataset construction, and experimental design can materially affect reported smart-contract detection performance [12]. Closely related or duplicated programs across training and test sets can also inflate apparent generalization [13], motivating the family-disjoint benchmark construction used in this work. Beyond clean evaluation, FinDet studies unseen attack patterns and feature obfuscation [6], while ContractShield and ORACAL evaluate vulnerability detection under adversarial perturbations [7], [8]. Recent work also examines structural and semantic obfuscation directly at the EVM-bytecode level [14]. These studies establish adversarial robustness as a relevant concern for learned smart-contract analysis, but they do not study score-aware adaptive screening at the EIP-7702 authorization boundary. More broadly,

adaptive attacks are a standard methodology for evaluating adversarial defenses because robustness measured against fixed transformations can fail once the attacker responds to the deployed model [15]. Malware research similarly demonstrates functionality-preserving black-box evasion through injection of nonexecuted content [16]. These observations are directly relevant to EIP-7702 because the party presenting delegate bytecode can also control its representation before authorization. AuthGuard-7702 therefore evaluates a score-aware adversary that searches over combinations of bytecode transformations rather than relying only on a predefined perturbation set.

Transformation validity creates a second challenge: problem-space evasion is meaningful only when the modified input retains task-relevant functionality [17]. Successful parsing or structural validity of EVM bytecode alone does not establish execution preservation. We therefore independently audit the primary transformation classes on a real EVM and separate broader transformations with weaker preservation evidence. Pre-authorization bytecode screening, score-aware adaptive evaluation, and execution-preservation auditing together distinguish AuthGuard-7702 from the approaches in Table I. Exploit generation in teEther [18], declarative decompilation in Gigahorse [19], and security analysis in Securify [20] offer complementary capabilities for subsequent analysis.

III. PROBLEM FORMULATION AND THREAT MODEL

We formulate EIP-7702 pre-authorization screening as binary risk screening of proposed delegate runtime bytecode. The screener warns of the analyzer-identified structural condition underlying the benchmark. We evaluate this decision against an adaptive author who can query the frozen screener and modify bytecode within a bounded transformation space.

A. Pre-Authorization Screening Task

Let b denote the normalized runtime bytecode of a proposed delegate, $T(b)$ its opcode-token representation, and $f_\theta(T(b))$ the corresponding uncalibrated logit. We convert this output into a calibrated screening score using validation-fitted temperature scaling [21]:

$$s(b) = \sigma\left(\frac{f_\theta(T(b))}{T_{\text{cal}}}\right), \quad T_{\text{cal}} > 0, \quad s(b) \in [0, 1], \quad (1)$$

Here $\sigma(\cdot)$ is the logistic sigmoid, T_{cal} is the validation-fitted temperature, and $s(b)$ is the resulting calibrated risk score.

Reference labels follow Huang et al. [9]. A delegate is *source-flagged* when the reference analysis identifies a decompiled reachability path from standardized inbound hooks, including `receive()`, `fallback()`, and token-reception callbacks, to predefined value-moving external-call patterns. Other delegates in the source population are *unflagged*. The screening target is therefore this analyzer-identified structural condition rather than malicious intent. Source-flagged and unflagged should not be interpreted as synonymous with malicious and safe, respectively. Operating thresholds are fitted exclusively on unflagged validation examples. Thresholds

$\tau_1 \geq \tau_5 \geq \tau_{10}$ correspond to nominal 1%, 5%, and 10% false-positive-rate operating points and define the warning levels. Adversarial experiments use τ_5 unless stated otherwise. The central robustness question is whether a screener continues to identify source-flagged delegates after the bytecode author adaptively modifies the program to suppress the warning.

B. Information Available at Authorization Time

The screener uses only the ordered opcode sequence derived from runtime bytecode available at the delegate address. `PUSH` immediates are excluded so that the representation captures opcode identity and order rather than embedded constants. No transaction history, balances, approvals, source code, address identity, reputation, call graphs, family identifiers, proxy state, simulation results, off-chain intent, or label-construction fields are provided to the model. This information boundary isolates the contribution of runtime bytecode to immediate pre-authorization screening and enables inference before richer contextual or semantic evidence is available.

C. Adversary Model

We consider an adversary who authors a delegate exhibiting the reference structural condition and seeks to avoid the screener’s warning before authorization. Starting from source bytecode b with $s(b) \geq \tau_5$, the adversary modifies the runtime bytecode and succeeds when an admissible transformed candidate falls below the fixed deployment threshold.

We grant the adversary three capabilities.

- **Score access.** The adversary can submit candidate bytecode to the frozen screener and observe the resulting score. Each search strategy receives at most 64 model queries per source delegate.
- **Bytecode modification.** The adversary can apply and compose transformations from the evaluated action space. The global attack space permits compositions of up to four actions, subject to tier-specific constraints, with at most one flooding action in any composition.
- **Bounded size overhead.** Appended content is limited to 200% of the original executable length, bounding a transformed candidate to approximately three times the original bytecode size.

The adversary does not receive model gradients or training access. This models a score-access screening interface in which the decision function can be probed before authorization. Stronger white-box attacks obtained through parameter recovery or direct access to an embedded model fall outside the evaluated threat model.

D. Evidence-Graded Attack Space

A security-relevant evasion must reduce the screening score without merely destroying the execution behavior that motivated the original warning. We therefore organize the attack space according to the strength of execution-preservation evidence and ground the primary robustness comparison in transformations supported by the real-EVM audit.

Algorithm 1 Score-aware adaptive attack

Require: Source bytecode b , frozen screener s , threshold τ_5 , action space $\mathcal{G}$, query budget $Q = 64$

Ensure: Best candidate b^* and attack outcome

```
1: if  $s(b) < \tau_5$  then
2:   return not attack-eligible
3: end if
4: for  $m \in \{\text{RANDOM}, \text{BEAM}\}$  do
5:   Initialize search state for  $m$  and  $b_m^* \leftarrow b$ 
6:    $q \leftarrow 0$ 
7:   while  $q < Q$  and candidates remain do
8:     Select the next admissible composition  $c$ 
9:      $b' \leftarrow c(b)$ 
10:    if  $b'$  is duplicate or violates attack constraints then
11:      continue
12:    end if
13:     $p \leftarrow s(b')$ ,  $q \leftarrow q + 1$ 
14:    Update the search state using  $p$ 
15:    if  $p < s(b_m^*)$  then
16:       $b_m^* \leftarrow b'$ 
17:    end if
18:  end while
19: end for
20:  $b^* \leftarrow \arg \min_{x \in \{b_{\text{Random}}^*, b_{\text{Beam}}^*\}} s(x)$ 
21: return  $(b^*, \mathbf{1}[s(b^*) < \tau_5])$ 
```

Tier I: Audit-Supported Flooding Oracle: Tier I appends a STOP followed by deterministic donor bytes equal to 25%, 50%, 100%, or 200% of the source executable length and selects the strongest of the four alternatives. Tier I provides a controlled flooding baseline without compositional adaptation.

Tier II: Audit-Supported Adaptive Adversary: Tier II adaptively composes flooding, metadata rewriting, and neutral instruction insertion. Each transformation class is supported by the real-EVM execution-preservation audit in Section VI-B1. Tier II therefore defines the primary adversary for the paper’s robustness claims.

Tier III: Broader Structural Stress Test: Tier III extends Tier II with rewriting of embedded addresses and function-selector immediates. Because these additional rewrites have substantially weaker preservation evidence, Tier III is analyzed separately as a broader stress test rather than used to support the primary robustness claim.

Tiers II and III use seeded random search and score-guided beam search under the same query budget and tier-specific action constraints. Algorithm 1 summarizes the attack for one source delegate and one frozen screener. Random search explores admissible transformation compositions in a seeded order. Beam search retains the four lowest-scoring candidates at each expansion step and follows the maximum composition depth permitted by the corresponding tier, up to depth four in Tier III. Random and beam search each receive an independent 64-query budget. Candidates must satisfy successful disassembly, opcode-skeleton validity, the byte-size limit, action compatibility, and the one-flood constraint. Duplicate or invalid candidates are removed before model scoring. Search continues after the first threshold crossing until the query budget is exhausted, and the lowest-scoring candidate produced by either strategy determines the final attack outcome.

Under Tier II, the four flooding levels together with metadata rewriting and neutral insertion define six audit-supported actions and admit 48 valid compositions under the one-flood constraint. Tier II compositions have maximum depth three, while the depth-four limit applies only to the broader Tier III space. After duplicate and validity filtering, the 64-query random procedure covers the complete defined Tier II composition space. The Tier II search is therefore exhaustive within the specified grammar rather than limited by incomplete exploration of that grammar. The preservation audit is conducted independently of Algorithm 1 and determines which transformation classes provide sufficient evidence to support the primary Tier II robustness claim.

E. Attack Success and Robust Recall

For a model operating at threshold τ_5 , an adversarial evasion occurs when a source-flagged delegate is detected before transformation but the selected transformed candidate is not:

$$s(b) \geq \tau_5 \quad \text{and} \quad s(b') < \tau_5, \quad (2)$$

Here b' is the strongest admissible transformed candidate generated from b under the evaluated attack. Attack success rate (ASR) is computed over source-flagged observations that the model detects on clean bytecode. It therefore measures the fraction of initial detections that can be successfully evaded. Because models can detect different clean populations, their marginal ASRs can have different denominators, motivating the common-population paired analysis defined in Section V. We additionally report *robust recall*, defined over all source-flagged observations as the fraction that are detected on clean bytecode and remain detected after the strongest evaluated attack. ASR measures susceptibility among initial detections, while robust recall captures the end-to-end fraction of the target population that survives adaptive modification.

IV. AUTHGUARD-7702

AuthGuard-7702 extracts a fast, bytecode-only risk signal from EIP-7702 delegates before an EOA grants them execution authority. We describe its pre-authorization workflow and AuthGuard-Seq architecture, motivated by the adaptive adversary in Section III.

A. Pre-Authorization Screening Workflow

Before authorization is signed, a wallet resolves the proposed delegate address and retrieves its runtime bytecode. Deterministic disassembly produces an opcode sequence for AuthGuard-Seq, whose logit is converted into a calibrated score by validation-fitted temperature scaling. Validation-derived thresholds determine the warning level. Training, family-disjoint partitioning, calibration, and threshold selection occur offline. The online path comprises bytecode retrieval, deterministic preprocessing, inference, and thresholding. More expensive semantic analysis, transaction simulation, and contextual checks remain available as complementary downstream evidence.

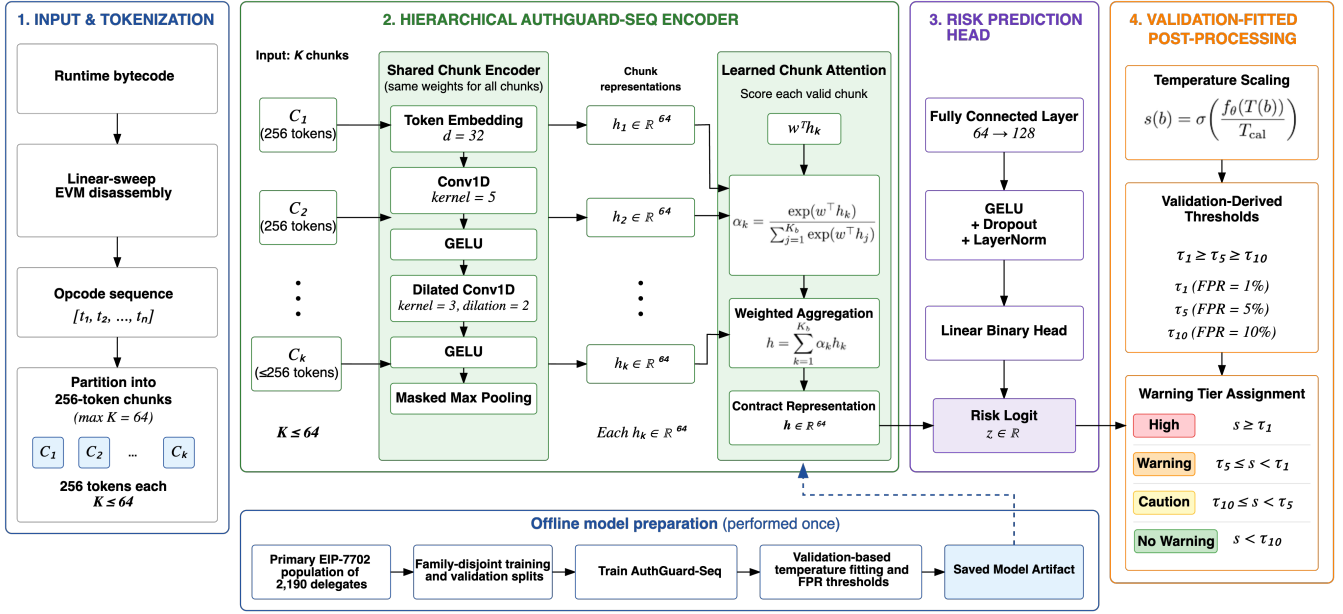


Fig. 2. AuthGuard-Seq architecture and validation-fitted post-processing

B. Design Rationale

The delegate author controls bytecode content and length. A useful representation must preserve local opcode structure, cover the full program within the benchmark regime, and prevent attacker-added regions from dominating solely through increased length. AuthGuard-Seq encodes contiguous opcode regions hierarchically using shared convolutions, then learns each region’s contribution to the contract-level representation. Under flooding and compositional attacks, appended content therefore does not automatically receive weight proportional to the number of added regions. The matched-control and attention-allocation experiments in Section VI-D identify selective chunk aggregation as the strongest architectural factor associated with robustness.

C. Hierarchical Opcode Representation

AuthGuard-Seq uses deterministic linear-sweep disassembly with stable opcode tokens, a dedicated undefined-byte token, separate padding, and `PUSH` immediates skipped according to instruction width. Disassembly covers the full runtime, including regions after the first linear `STOP`, keeping trailing regions visible to both screener and adversary. The sequence is divided into 256-token chunks, at most $K = 64$, for a capacity of 16,384 opcodes. The longest primary benchmark program has 16,081 opcodes, so none is truncated. Each chunk uses 32-dimensional opcode embeddings and shared one-dimensional convolutions with kernel sizes 5 and 3 (second-layer dilation 2). Masked max pooling produces a 64-dimensional regional representation h_k per chunk.

D. Selective Chunk Aggregation

AuthGuard-Seq learns the contribution of each non-padding chunk to the contract-level representation. For a contract con-

taining K_b valid chunks, the attention weights and aggregated representation are:

$$\alpha_k = \frac{\exp(w^\top h_k)}{\sum_{j=1}^{K_b} \exp(w^\top h_j)}, \quad h = \sum_{k=1}^{K_b} \alpha_k h_k, \quad (3)$$

Here $h_k \in \mathbb{R}^{64}$ denotes the representation of chunk k , $w \in \mathbb{R}^{64}$ is the learned attention vector, α_k is the normalized contribution assigned to chunk k , and h is the resulting contract-level representation. Unlike uniform aggregation, this mechanism allows the influence of a region to depend on its encoded content rather than on program length alone. This property is important under bytecode flooding because newly appended regions are not guaranteed a proportional share of the final representation. The aggregated representation passes through a 128-dimensional hidden layer with GELU activation, dropout ($p = 0.15$), and layer normalization, followed by a binary classification head. The resulting logit is calibrated using the validation procedure described in Section V. Figure 2 summarizes the complete architecture.

E. Model Capacity and Evaluation Scope

The reference AuthGuard-Seq has 63,266 active parameters, versus 108,225 for the neural n -gram and 75,595 for the dense structural model in clean evaluation. Its contribution is organizing bytecode representation for adaptive pre-authorization screening rather than increasing capacity alone. Section VI-D tests this interpretation using parameter-matched controls and attention-allocation analysis, separating the effect of selective aggregation from parameter count and clean predictive performance.

V. BENCHMARK AND METHODOLOGY

This section describes benchmark construction, family-disjoint partitioning, training, calibration, and statistical procedures for clean and adversarial evaluation.

TABLE II
BENCHMARK POPULATIONS AND OPERATIONAL EVALUATION RESOURCES

Population	n	Label / status
Primary evaluation	2,190	727 flagged / 1,463 unflagged
External benign control	797	Benign-labeled
Curated benchmark controls	5	Curated legitimate implementations
Excluded unusable input	90	Unusable runtime bytecode
Benchmark total	3,082	
<i>Additional resources used for operational evaluation</i>		
Audited-implementation registry	45	8 projects, 30 unique bytecodes
Later temporal population	752	Unlabeled, 669 unique bytecodes

A. Benchmark Construction

We construct *AuthGuardBench-7702* from the populations summarized in Table II. The primary evaluation population is derived from the EIP-7702 delegate dataset and analyzer outputs of Huang et al. [9] and contains 2,190 delegates, comprising 727 source-flagged and 1,463 unflagged examples. To evaluate behavior outside this population, the benchmark also includes 797 benign-labeled Ethereum contracts from PhishingHook [4] and five curated production EIP-7702 implementations. Data auditing identified 90 records whose original verdict could not be associated with usable runtime bytecode. Among these, 89 unflagged records were truncated by a spreadsheet cell limit, and one record contained a timeout string in place of bytecode. These records are retained only in corpus accounting and are excluded from model evaluation. The benchmark therefore contains 3,082 records in total, of which 2,992 have usable runtime bytecode. We additionally maintain two resources exclusively for operational evaluation. The audited-implementation registry contains 45 deployment records from eight EIP-7702 and account-abstraction projects, corresponding to 30 unique runtime bytecodes. We also collect a later temporal population of 752 screenable delegates. These resources remain separate from the benchmark and are used only for the operational analyses in Section VI-E. Neither contributes to training, model selection, calibration, or the primary detection results.

B. Family-Disjoint Partitioning

To prevent leakage from duplicated or closely related programs, we construct bytecode families before partitioning the benchmark. Smart-contract code is frequently reused [22], and duplicated programs across training and test sets can inflate learned-code evaluation [13]. More generally, unrealistic train and test partitions can bias security-ML evaluation [23]. Runtime bytecode is therefore linearly disassembled and represented as sets of opcode 4-grams. Following the resemblance-estimation principle underlying MinHash [24], we construct deterministic signatures using 128 BLAKE2b-based permutations. Pairs with signature agreement of at least 0.85 are connected, and connected components define bytecode families. This procedure yields 790 families in the primary evaluation population. Identical runtime bytecodes with conflicting labels are quarantined before modeling.

We construct five fixed outer folds at the family level. For outer fold f , fold f is used for testing, fold $(f + 1) \bmod 5$

for validation, and the remaining three folds for training. As a result, identical bytecode and members of the same similarity family cannot cross training, validation, and test partitions. The primary evaluation uses these frozen 0.85 family-disjoint folds. Section VI-B separately evaluates sensitivity to the family definition by rebuilding family-disjoint partitions at MinHash thresholds 0.80, 0.85, and 0.90. A dedicated bridge evaluation links the threshold-specific 0.85 analysis to the frozen primary benchmark. Family membership is fixed across training seeds, so changing a seed affects model optimization and attack search rather than the benchmark partition.

C. Models, Training, and Calibration

The clean evaluation compares seven predefined model configurations under the same family-disjoint folds and validation rotation: AuthGuard-Seq, a flat CNN, histogram plus opcode-4-gram XGBoost [25], a neural n -gram model, BiGRU, a dense structural model, and a Transformer. These configurations cover sequence, frequency, recurrent, structural, and attention-based representations rather than restricting the comparison to close variants of AuthGuard-Seq. We additionally use an L2-regularized, class-balanced logistic-regression screener ($C = 1.0$) as the central robustness reference. The LR operates on 15 hand-designed opcode and control-flow features capturing code size, opcode diversity, external-call indicators, dispatcher and fallback structure, hard-coded address evidence, and conservative fallback-to-external-call reachability. Its intentionally simple and interpretable design provides a useful reference for determining whether additional model complexity translates into greater robustness under adaptive attack.

Neural configurations use class-weighted binary cross-entropy and AdamW with learning rate 10^{-3} , weight decay 10^{-4} , and gradients clipped at 5. Training lasts at most 30 epochs with validation-AUPRC early stopping (patience 5); the best validation-AUPRC checkpoint is restored. Temperature scaling uses only validation logits, and nominal 1%, 5%, and 10% thresholds use unflagged validation examples. Test labels never inform training, model selection, calibration, or threshold fitting. For each model, seed, and fold, Tiers I, II, and III share the frozen checkpoint, calibration temperature, and thresholds, without retraining or recalibration. Experiments use seeds 7702, 7703, and 7704 where replication permits, while Section VI identifies single-seed supporting experiments.

D. Statistical Analysis

Bootstrap comparisons sample bytecode families with replacement, preserving dependence among related contracts. Paired analyses apply the same sampled family multiplicities to both models. We use 10,000 replicates for principal clean and adversarial comparisons, 2,000 for the mechanism ablation, and 1,998 for supporting baseline comparisons. Marginal attack success rate (ASR) measures successful warning removal among source-flagged observations detected on clean bytecode by a given model. Different models may detect different clean populations, so marginal ASRs cannot provide a direct paired comparison. Paired inference instead compares attack outcomes

on source-flagged observations detected cleanly by both models. The paired ASR difference is estimated directly on this shared population, rather than by subtracting marginal ASRs. Robust recall uses all source-flagged observations as the denominator, accounting jointly for clean misses and successful adaptive evasions.

VI. EVALUATION

We evaluate AuthGuard-7702 through five research questions spanning clean predictive performance, execution-audited adaptive robustness, architectural mechanism, and operational behavior.

- **RQ1:** How do candidate screeners compare under clean evaluation?
- **RQ2:** How robust are AuthGuard-Seq and LR under an execution-audited adaptive adversary?
- **RQ3:** Does a broader action space with weaker preservation evidence materially change the robustness conclusion?
- **RQ4:** Which architectural factor is most strongly associated with the observed robustness?
- **RQ5:** How does AuthGuard-7702 behave on external data and under deployment constraints?

A. RQ1: How Do Candidate Screeners Compare Under Clean Evaluation?

We begin with the conventional setting in which test bytecode is not modified by an adversary. Table III reports family-disjoint performance across three seeds and five folds. AuthGuard-Seq achieves the highest mean performance, with AUPRC 0.924 ± 0.014 and Recall@5% 0.833 ± 0.016 . In a separate paired comparison, however, the 15-feature LR screener reaches 0.911 AUPRC, close to AuthGuard-Seq’s 0.924. The family-clustered LR-minus-AuthGuard difference is -0.021 with a 95% CI of $[-0.078, +0.035]$, so clean AUPRC does not statistically separate the two screeners. By contrast, MinHash k NN reaches only 0.612 AUPRC, with a paired difference from AuthGuard-Seq of -0.302 ($[-0.380, -0.226]$), providing evidence against trivial nearest-neighbor memorization.

B. RQ2: Robustness Under an Execution-Audited Adversary

We test the two principal screeners against active evasion, separating preservation from attack effectiveness. After establishing independent execution-preservation evidence for the primary transformation classes, we apply Algorithm 1’s budgeted search to frozen checkpoints and operating thresholds. Attack success is thus interpreted alongside preservation of the measured execution behavior.

1) *Execution-Preservation Audit:* We audit the primary transformations on a real EVM using 120 delegates from distinct families. For each delegate, original and transformed runtime bytecode are executed under the same harness on empty calldata, the zero selector, and up to eight selectors recovered from the program. Preservation requires agreement on failure status, return data, the set and count of external calls, storage writes, and log count. Across flooding, metadata rewriting, and neutral insertion, all 5,196 transformed executions preserve the measured behavior over 120 bytecode families. For flooding,

TABLE III
CLEAN FAMILY-DISJOINT PERFORMANCE ACROSS THREE SEEDS AND FIVE FOLDS

Model	Params	AUPRC $\uparrow$	AUROC $\uparrow$	Brier $\downarrow$	R@5% $\uparrow$
AuthGuard-Seq	63,266	.924$\pm$.014	.963$\pm$.011	.072$\pm$.012	.833$\pm$.016
Flat CNN	154,177	.885 $\pm$.010	.937 $\pm$.007	.099 $\pm$.004	.712 $\pm$.024
Hist.+4-gram XGB	—	.833 $\pm$.004	.907 $\pm$.002	.127 $\pm$.004	.615 $\pm$.015
Neural n -gram	108,225	.810 $\pm$.007	.880 $\pm$.020	.125 $\pm$.001	.654 $\pm$.029
BiGRU	108,225	.679 $\pm$.098	.815 $\pm$.069	.163 $\pm$.027	.379 $\pm$.113
Dense structural	75,595	.637 $\pm$.018	.780 $\pm$.022	.182 $\pm$.009	.331 $\pm$.023
Transformer	425,217	.563 $\pm$.031	.730 $\pm$.019	.208 $\pm$.013	.239 $\pm$.054

TABLE IV
REAL-EVM EXECUTION-PRESERVATION AUDIT

Transformation class	Families	Execs.	Preserved	Wilson 95% CI
<i>Tiers I and II, audit-supported</i>				
Flooding (25–200%)	120	3,464	3,464	[0.9989, 1.0000]
Metadata rewrite	120	866	866	[0.9956, 1.0000]
Neutral insertion	120	866	866	[0.9956, 1.0000]
<i>Tier III only, weaker evidence</i>				
Address rewrite (not promoted)	10	100	23	[0.1584, 0.3215]

the audit includes 480 empty-calldata executions and 480 zero-selector executions, and 788 audited flooding cases have an original execution that reaches an external call. These results provide execution-level preservation evidence over the audited calldata and observables. Address rewriting preserves only 23 of 100 audited executions, while selector rewriting is not included in the execution-preservation audit. Tier III is therefore evaluated separately as a broader structural stress test rather than as evidence supporting the primary robustness claim. Table IV summarizes the audit.

2) *Attack Success Across Tiers:* Table V reports marginal ASR and robust recall, with ASR denominators comprising each model’s own cleanly detected positives. AuthGuard-Seq and the 15-feature LR baseline use all three seeds. The Flat CNN and the gradient-boosted model use only seed 7702 and provide supporting context, not replication-matched ranking. Under the primary Tier II adversary, marginal ASR is 0.197 for AuthGuard-Seq versus 0.542 for LR. Because these denominators differ, the paired analysis below directly estimates the models’ robustness difference.

3) *Central Paired Comparison:* Direct comparison uses 1,503 source-flagged observations detected cleanly by both models, from 209 bytecode families across three seeds, and asks which screener better preserves its warning after adaptive modification (Table VI). Under the primary Tier II adversary, LR attack success exceeds AuthGuard-Seq by 0.3699, with a family-clustered 95% confidence interval of $[+0.208, +0.520]$. The screeners, though not statistically separated by clean AUPRC, therefore exhibit a substantial, statistically supported adaptive robustness difference.

Family-threshold sensitivity: To test whether this result depends on the 0.85 MinHash family threshold, we rebuild the family-disjoint partitions over the same 2,190 observations at thresholds 0.80, 0.85, and 0.90. These settings produce 701, 790, and 928 families, respectively. Table VII shows that the paired Tier II advantage remains positive at all

TABLE V

MARGINAL ATTACK SUCCESS AND ROBUST RECALL ACROSS ATTACK TIERS

Model	Tier	ASR	95% CI	Robust recall	Seeds
AuthGuard-Seq	I	.1492	[.105, .203]	.7189	3
	II	.1970	[.148, .256]	.6786	3
	III	.2084	[.158, .268]	.6690	3
15-feature LR baseline	I	.3083	[.218, .413]	.5708	3
	II	.5417	[.427, .645]	.3783	3
	III	.5722	[.461, .675]	.3530	3
Flat CNN	I	.8109	[.718, .889]	.1293	1
	II	.9356	[.893, .972]	.0440	1
	III	.9477	[.914, .978]	.0358	1
Hist.+4-gram XGB	I	.8111	[.754, .866]	.1169	1
	II	.9844	[.967, .997]	.0096	1
	III	.9911	[.981, .998]	.0055	1

three thresholds, with family-clustered confidence intervals above zero. The magnitude changes as the definition of a bytecode family changes, but the robustness ordering does not reverse. Each threshold uses its corresponding rebuilt family-disjoint partition. At 0.85, the rebuilt analysis yields a +0.365 paired advantage. A bridge evaluated on the frozen benchmark split yields +0.366 ([+0.211, +0.518]), closely matching the primary +0.370 ([+0.208, +0.520]) result.

4) *End-to-End Robust Recall*: Tier II robust recall is 0.6786 for AuthGuard-Seq compared with 0.3783 for LR. The single-seed Flat CNN and XGBoost retain 0.0440 and 0.0096, respectively, and are reported only as supporting baselines. Together with the paired 0.370 ASR gap, robust recall shows that AuthGuard-Seq retains substantially more of the source-flagged population after adaptive attack as well as outperforming LR on their common clean-detected population.

C. RQ3: What Does the Broader Action Space Add?

Expanding the attack space from Tier II to Tier III changes the paired AuthGuard-Seq versus LR difference only from +0.3699 to +0.3826. The central robustness separation is therefore already present within the execution-audited Tier II action space and is not driven by the weaker-preservation Tier III rewrites. The larger change occurs between the flooding-only Tier I adversary, which yields +0.2083 (95% CI [+0.076, +0.334]), and the score-aware Tier II adversary, which reaches +0.3699 ([+0.208, +0.520]). Adaptive composition over the audited transformations therefore exposes substantially more separation than flooding alone.

D. RQ4: What Explains the Robustness Difference?

We next test whether selective aggregation, rather than model size or full-stream coverage alone, is associated with the robustness difference.

1) *Parameter-Matched Controls Under Flooding*: We compare approximately parameter-matched controls to separate model capacity, full-stream coverage, and aggregation as possible explanations. Giving a flat encoder the same 16,384-token budget produces a clean hierarchy contrast of -0.018 AUPRC with a 95% CI of $[-0.040, +0.018]$, so clean performance does not establish hierarchical coverage alone as the source of the robustness advantage. Under flooding,

TABLE VI

PAIRED ATTACK-SUCCESS COMPARISON BETWEEN THE LR BASELINE AND AUTHGUARD-SEQ

Tier	Adversary	LR ASR	AuthGuard ASR	Δ ASR	95% CI
I	flooding oracle	.3353	.1271	+ .2083	[+.076, +.334]
II	audited adaptive	.5369	.1670	+ .3699	[+.208, +.520]
III	full stress test	.5635	.1810	+ .3826	[+.223, +.524]

TABLE VII

MINHASH-THRESHOLD SENSITIVITY OF PAIRED TIER II ROBUSTNESS

MinHash threshold	Families	Δ ASR (LR-AG)	[95% CI]
0.80	701	+ .423	[+.315, +.525]
0.85	790	+ .365	[+.164, +.551]
0.90	928	+ .254	[+.144, +.352]

however, chunk attention exceeds chunk mean pooling by +0.180 AUPRC ([+0.135, +0.232]) and +0.541 Recall@5% ([+0.474, +0.631]). This contrast motivates a direct test of the aggregation mechanism under the primary adaptive adversary.

2) Parameter-Matched Controls Under Adaptive Attack:

We next test whether the aggregation result persists under the primary score-aware Tier II adversary. Table VIII reports the parameter-matched experiment across seeds 7702, 7703, and 7704. The attacker uses the audit-supported Tier II transformations of flooding, metadata rewriting, and neutral instruction insertion under the same 64-query-per-strategy budget and Tier II composition limits used in RQ2. Across all three seeds and both search strategies, chunk attention remains substantially harder to evade. On the common clean-detected population, seed-averaged random-search differences are +0.881 (95% CI [+0.813, +0.933]) for mean-minus-attention and +0.779 ([+0.715, +0.836]) for flat-minus-attention. Beam-search differences are +0.848 ([+0.795, +0.894]) and +0.649 ([+0.584, +0.710]), respectively. All 12 per-seed paired confidence intervals exclude zero. The attention-versus-mean comparison most directly isolates selective aggregation because both variants share the chunk encoder, 16,384-token budget, and hierarchical representation and differ only in learned versus uniform aggregation plus 64 trainable parameters. The flat control also changes the representation layout and therefore provides supporting rather than isolating evidence. The same aggregation ordering is observed across all three seeds.

3) *Where Does the Model Place Its Attention?*: We further examine the proposed dilution mechanism by measuring how the trained aggregator distributes attention as attacker-added content increases. For each of the 727 source-flagged delegates, evaluated in the outer fold where it is held out, we measure the attention assigned to chunks composed entirely of appended donor content and compare it with the mass those chunks would receive under uniform weighting. Under Flood-200%, appended content accounts for 63.1% of all chunks but receives only 37.6% of the learned attention mass. The gap between appended chunk share and learned attention grows from 8.2 percentage points at Flood-25% to approximately 26 points at Flood-200%. Uniform weighting is an analytic reference rather than a separately trained model, so this allocation analysis is

TABLE VIII
PARAMETER-MATCHED TIER II CONTROLS ACROSS THREE SEEDS

Configuration	Params	Clean recall	Random ASR	Beam ASR
Chunk attention	30,049	.869 $\pm$.042	.092$\pm$.005	.069$\pm$.006
Flat single window	29,985	.819 $\pm$.008	.867 $\pm$.031	.716 $\pm$.043
Chunk mean pooling	29,985	.783 $\pm$.006	.982 $\pm$.001	.923 $\pm$.009

ASR columns report marginal values. Paired differences in the text are estimated on the common clean-detected population.

interpreted together with the parameter-matched controls rather than as an independent causal ablation. The two analyses provide complementary evidence that selective aggregation suppresses dilution from attacker-added content within the evaluated model family. Figure 3 shows how this allocation gap develops as flooding increases.

E. RQ5: How Does AuthGuard-7702 Behave Operationally?

External and production controls: Applying validation thresholds to 797 benign-labeled general Ethereum contracts yields false-positive rates of 0.015 ± 0.004 , 0.065 ± 0.012 , and 0.169 ± 0.021 at nominal 1%, 5%, and 10% operating points. The upward shift reaches 16.9% at the nominal 10% point, showing that validation-fitted thresholds do not transfer directly to this external population. In the separate audited-implementation registry, three of eight named projects overlap primary training families and are excluded from aggregate interpretation. None of the five leakage-clean projects is flagged at the 5% point. Given the small sample, this control is used as a deployment sanity check rather than an estimate of production false-positive rate.

Later delegation traffic: We scan 100,444 blocks containing 1,335,391 authorization entries and identify 752 distinct screenable delegates over a two-week window, corresponding to 669 unique bytecodes. No exact bytecode hash overlaps the benchmark, although the 0.85 MinHash criterion places 70 delegates (9.3%) in a training family, leaving 682 leakage-clean delegates. Because the population is unlabeled, flag rates characterize score transfer rather than detection accuracy. At the nominal 5% point, 20.3% of delegates are flagged and 19.9% of leakage-clean delegates are flagged. At the 1% point, the corresponding rates are 9.5% and 8.8%. Weighted by authorization volume, flagged delegates account for 9.4% of volume at the 5% point and 5.9% at the 1% point. These shifts show that benchmark-derived operating thresholds require deployment-specific monitoring and recalibration.

Latency and staged analysis: Across 1,500 complete screening calls on a single CPU thread, AuthGuard-7702 has median latency 4.121 ms, p95 latency 14.547 ms, and p99 latency 21.429 ms. A pinned Gigahorse [19] decompiler image completes 60 family-distinct inputs with a warm median of 2.687 s and p95 latency 5.618 s, approximately $652\times$ the median AuthGuard-7702 screening latency. This comparison characterizes analysis latency rather than detector accuracy because the Gigahorse measurement covers decompilation and lifting and is not the Huang et al. analyzer. The 15-feature LR baseline has a median latency of 2.626 ms. These measurements place AuthGuard-Seq in the millisecond-scale regime

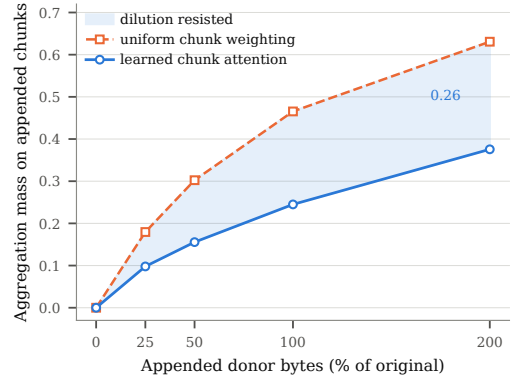


Fig. 3. Aggregation mass assigned to appended donor chunks under flooding

compatible with synchronous pre-authorization screening while leaving substantially more expensive analysis stages available for delegates that require deeper inspection.

VII. DISCUSSION

We discuss security implications, architectural factors associated with robustness, deployment, and the study’s boundaries.

A. Security and Model-Selection Implications

Clean and adaptive evaluation lead to different model-selection conclusions in the EIP-7702 setting. AuthGuard-Seq and the 15-feature LR screener are not statistically separated by clean AUPRC, yet a substantial robustness gap emerges once the delegate author adapts bytecode against the frozen screening decision. For attacker-authored programs, adaptive robustness is therefore a model-selection property rather than a secondary stress test. The preservation audit strengthens this conclusion by separating meaningful evasion from transformations that merely disrupt execution. Tier II grounds the primary comparison in three audit-supported transformation classes, and the 64-query random procedure covers its complete defined grammar. The robustness advantage also remains positive across MinHash thresholds of 0.80, 0.85, and 0.90. Together, these checks show that the result persists across the primary audited attack space and reasonable alternatives for constructing bytecode families.

B. Architectural Insights and Deployment Implications

The mechanism analysis identifies selective chunk aggregation as the strongest explanation for the robustness advantage within the evaluated architecture. Parameter-matched adaptive controls show that chunk attention is substantially harder to evade than uniform chunk averaging, while the attention-allocation analysis shows that attacker-added regions receive less representation mass than under uniform weighting. Because the attention-versus-mean comparison holds the hierarchical encoder and token budget fixed, the two analyses provide converging evidence that selective aggregation helps resist dilution from attacker-added bytecode.

AuthGuard-Seq’s millisecond-scale latency supports synchronous screening before authorization. Its score can prioritize delegates for deeper semantic analysis, transaction simulation,

reputation checks, or manual review. The later-traffic experiment also shows that benchmark-fitted thresholds shift under deployment populations, so score monitoring and threshold recalibration remain important in deployment.

C. Limitations and Threats to Validity

The study’s conclusions are bounded by the analyzer-defined structural target, the audited calldata and observables, and the defined transformation grammar, depth, query budget, and size overhead. The results therefore concern detection and comparative robustness for this target rather than general malicious-contract classification or complete semantic equivalence. Model extraction and white-box attacks remain outside the evaluated threat model. The main AuthGuard-Seq and LR comparison and matched aggregation controls use three seeds, while the supporting Flat CNN and XGBoost tier experiments use seed 7702 only. The family-threshold analysis preserves a positive Tier II advantage across three alternative family constructions, although other similarity definitions remain untested. Finally, the later population of 752 delegates is unlabeled and spans two weeks, so it measures score and operating-point transfer rather than field detection accuracy. Longer deployment periods, additional chains, and richer semantics-preserving attack spaces are natural extensions.

VIII. CONCLUSION

EIP-7702 creates a new security decision before an externally owned account authorizes delegate code, and this work studies how that decision can be supported when the delegate author may actively modify the bytecode to evade screening. Clean AUPRC does not statistically separate AuthGuard-Seq from the 15-feature LR screener, but the primary execution-audited Tier II adversary reveals a substantial difference. Paired attack success is 0.167 for AuthGuard-Seq versus 0.537 for LR, and robust recall is 0.679 versus 0.378. This advantage remains positive across alternative bytecode-family definitions. A real-EVM audit supports the primary attack space with 5,196 transformed executions from 120 bytecode families. Parameter-matched controls and attention-allocation analysis identify selective chunk aggregation as the main factor associated with robustness. AuthGuard-7702 thus shows that adaptive, preservation-aware evaluation reveals security differences hidden by clean testing and provides a practical bytecode-based approach to EIP-7702 pre-authorization screening.

REFERENCES

- [1] V. Buterin, S. Wilson, A. Dietrichs, and lightclient, “EIP-7702: Set code for EOAs,” Ethereum Improvement Proposals, May 2024. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-7702>
- [2] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, “ContractWard: Automated vulnerability detection models for Ethereum smart contracts,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1133–1144, Apr.–Jun. 2021.
- [3] N. Ashizawa, N. Yanai, J. P. Cruz, and S. Okamura, “Eth2Vec: Learning contract-wide code representations for vulnerability detection on Ethereum smart contracts,” *Blockchain: Research and Applications*, vol. 3, no. 4, p. 100101, Dec. 2022.
- [4] P. De Rosa, S. Queyrut, Y.-D. Bromberg, P. Felber, and V. Schiavoni, “PhishingHook: Catching phishing Ethereum smart contracts leveraging EVM opcodes,” in *Proc. 55th Annu. IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN)*, Naples, Italy, 2025, pp. 222–232.
- [5] T. Abdelaziz and A. Hobor, “Smart learning to find dumb contracts,” in *Proc. 32nd USENIX Security Symposium*, Anaheim, CA, USA, 2023, pp. 1775–1792.
- [6] Y. Liu, X. Su, H. Wu, S. Li, Y. Cheng, F. Xu, and S. Zhong, “Revealing adversarial smart contracts through semantic interpretation and uncertainty estimation,” *arXiv preprint arXiv:2509.18934*, 2025.
- [7] M.-D. Tran-Duong, N. H. Phong, N. C. Thanh, D. M. Trung, T. Truong-Huu, V.-H. Pham, and P. T. Duy, “ContractShield: Bridging semantic-structural gaps via hierarchical cross-modal fusion for multi-label vulnerability detection in obfuscated smart contracts,” *arXiv preprint arXiv:2604.02771*, 2026.
- [8] T. D. M. Dai, T. H. M. Le, M. A. Babar, V.-H. Pham, and P. T. Duy, “ORACAL: A robust and explainable multimodal framework for smart contract vulnerability detection with causal graph enrichment,” *arXiv preprint arXiv:2603.28128*, 2026.
- [9] M. Huang, H. Liu, S. Yang, D. Wu, and S. Wang, “Revealing the dark side of smart accounts: An empirical study of EIP-7702 incurred risks in blockchain ecosystem,” in *Proc. 35th USENIX Security Symposium*, Baltimore, MD, USA: USENIX Association, Aug. 2026, pp. 1887–1905.
- [10] M. Qi, Q. Wang, R. Li, T. Zhu, and S. Chen, “EIP-7702 phishing attack,” *arXiv preprint arXiv:2512.12174*, Dec. 2025.
- [11] B. Zhang, N. He, X. Hu, K. Ma, and H. Wang, “Following devils’ footprint: Towards real-time detection of price manipulation attacks,” in *Proc. 34th USENIX Security Symposium*, Seattle, WA, USA, 2025, pp. 4127–4145.
- [12] B. Wang, Y. Tong, S. Ji, H. Dong, X. Luo, and P. Zhang, “A review of learning-based smart contract vulnerability detection: A perspective on code representation,” *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 6, pp. 1–46, May 2026, article 167.
- [13] M. Allamanis, “The adverse effects of code duplication in machine learning models of code,” in *Proc. ACM Onward!*, 2019, pp. 143–153.
- [14] P. Zhang, X. Wang, Y. Tong, H. Dong, Y. Xiao, and S. Ji, “Bytecode obfuscation for smart contracts via structural and semantic perturbation,” *ACM Transactions on Software Engineering and Methodology*, 2026.
- [15] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1633–1645.
- [16] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, “Functionality-preserving black-box optimization of adversarial windows malware,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3469–3478, 2021.
- [17] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, “Intriguing properties of adversarial ML attacks in the problem space,” in *Proc. IEEE Symp. Security and Privacy (SP)*, San Francisco, CA, USA, 2020, pp. 1332–1349.
- [18] J. Krupp and C. Rossow, “teEther: Gnawing at Ethereum to automatically exploit smart contracts,” in *Proc. 27th USENIX Security Symposium*, Baltimore, MD, USA, 2018, pp. 1317–1333.
- [19] N. Grech, L. Brent, B. Scholz, and Y. Smaragdakis, “Gigahorse: Thorough, declarative decompilation of smart contracts,” in *Proc. IEEE/ACM 41st Int. Conf. Software Engineering (ICSE)*, Montreal, QC, Canada, 2019, pp. 1176–1186.
- [20] P. Tsankov, A. Dan, D. Drachler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, “Securify: Practical security analysis of smart contracts,” in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Toronto, ON, Canada, 2018, pp. 67–82.
- [21] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Proc. ICML*, 2017, pp. 1321–1330.
- [22] X. Chen, P. Liao, Y. Zhang, Y. Huang, and Z. Zheng, “Understanding code reuse in smart contracts,” in *Proc. IEEE Int. Conf. Software Analysis, Evolution and Reengineering (SANER)*, 2021, pp. 470–479.
- [23] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: Eliminating experimental bias in malware classification across space and time,” in *Proc. 28th USENIX Security Symposium*, Santa Clara, CA, USA, 2019, pp. 729–746.
- [24] A. Z. Broder, “On the resemblance and containment of documents,” in *Proc. Compression and Complexity of Sequences*, 1997, pp. 21–29.
- [25] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.