Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 47 No. 03 (2026)

Predictive Moving Target Defense for Ransomware Using Temporal Transformers and Uncertainty-Aware Risk Estimation

Snehal R. Shinde[1], Dr. Jagdish W. Bakal[2]

[1]PhD Schlor, Pillai HOC College of Engineering and Technology, Rasayani, Maharashtra, India

[2]Principal, Pillai HOC College of Engineering and Technology, Rasayani, Maharashtra, India

Abstract

Ransomware has evolved into one of the most disruptive cyber threats, leading to prolonged service outages and irreversible data loss. Conventional static defense mechanisms and signature-based detection methods are no longer sufficient to counter adaptive attack strategies and rapidly evolving ransomware variants. Moving Target Defense (MTD) has attracted considerable interest as a proactive defense paradigm by continuously varying system configurations for increasing attacker uncertainty. However, existing MTD mechanisms are often reactive, rule-based, or weakly synchronized with detection engines, resulting in unnecessary and potentially disruptive reconfigurations, delayed responses, and degraded system performance. To address these limitations, a machine learning–driven, behaviour-aware MTD framework is proposed for ransomware defense. The behavioral traces are processed through sliding-window analysis and hybrid feature encoding that integrates statistical descriptors with categorical embeddings. A time-series Transformer encoder models temporal attack patterns, whereas uncertainty-aware risk estimation, together with Monte Carlo Dropout, facilitates confidence-guided MTD triggering. Defense actions are activated only when predicted risk surpasses predefined thresholds with low uncertainty, minimizing false alarms and redundant system reconfigurations. Moreover, a feedback-driven self-supervised adaptive learning mechanism enables the model to handle behavioural drift over time. The MLRan dataset is used for simulations, which demonstrate high ransomware detection reliability with timely MTD activation, reduced false positives, and improved system resilience, proving the efficacy of the proposed framework for real-time capable ransomware mitigation.

Keywords: Ransomware Detection, Moving Target Defense, Time-Series Transformer, Uncertanity-Aware Risk Estimation, Behavioral Modeling, Monte Carlo Dropout, Self- Supervised Learning, Real-Time Threat Mitigation

1. Introduction

Ransomware has increasingly been regarded as one of the most pervasive and destructive forms of cybercrime that can encrypt critical data and coerce victims enterprises, individuals, and even governments into paying ransom for decryption [1]. Over the past decade, the ransomware ecosystem has extended rapidly in scale, sophistication, and automation, causing billions of dollars in global damage. With almost 20,000 new ransomware variants emerging each month, conventional defenses confront an unsustainable burden due to tracking obfuscated payloads, continuously updating signatures, and analyzing ever-evolving malicious behaviors [2]. Ransomware can escape pattern-matching through dynamic API generation, code obfuscation, encryption of strings associated with cryptocurrencies, and anti-analysis methods like sandbox detection and process hollowing. Additionally, the file I/O patterns of ransomware reflect benign software, making API-level monitoring susceptible to false positives [3].

This widening gap has motivated the cybersecurity community to explore adaptive, proactive, and uncertainty-driven strategies. Among these strategies, Moving Target Defense (MTD) has emerged as a leading paradigm

shift from static fortification to dynamic, attack-surface transformations [4]. MTD presents unpredictability by continuously changing system configurations like file attributes, memory layouts, network parameters, APIs, or execution environments to invalidate attackers' reconnaissance and increase their operational cost. Techniques including IP hopping, address space layout randomization, and mutable network topologies prove the power of shuffling, diversity, and redundancy in impeding attackers' capacity to execute deterministic attacks [5]. In ransomware, MTD enables defenders to interrupt core phases of the attack chain, especially the file-search and encryption phases, by dynamically changing directory structures, shuffling file extensions, or modifying access patterns. Even a single shuffle can prevent ransomware that depends on static file-type targeting, and repeated shuffling can develop a robust, low-cost defense surface [6][7].

Despite its potential, research on MTD remains fragmented and largely focused on the network layer, with limited exploration of data-layer and file-system dynamics central to ransomware behaviour. Recent developments increasingly highlight intelligent, self-adaptive MTD mechanisms driven by advanced machine learning (ML), especially Transformer-based architectures, which learn complex temporal and behavioural patterns to optimize when, where, and how the attack surface must be dynamically reconfigured in response to ransomware activity [8][9]. However, ML-enabled intelligent MTD remains at an early stage, with limited unified frameworks and insufficient alignment with ransomware-specific behaviours like privilege escalation, staged encryption, and lateral movement.

1.1 Paper Organization

The rest of this paper is organized as follows. Section 2 surveys some promising solutions for MTD techniques based on ML techniques. Section 3 presents the attack model, system model, and problem formulation, whereas Section 4 details the proposed ML-based MTD framework. Section 5 describes the simulation environment, evaluation metrics, and baselines, and provides results and discussion. Finally, Section 6 concludes the paper and outlines future research directions.

2. Literature Review

This section reviews MTD approaches targeting ransomware attacks and ML–based MTD frameworks that improve adaptive and intelligent defense capabilities.

2.1 Moving Target Defense Approaches Against Ransomware

Several existing studies have explored MTD as a proactive countermeasure to alter the attack surface to interrupt ransomware execution dynamically. Four lightweight MTD mechanisms are proposed in [10] that span network, data, and runtime layers, coordinated by an IoT-oriented decision framework. It concentrates on IoT environments in which limited resources and multi-purpose malware exacerbate security challenges. To tackle data-theft ransomware, a Windows-centric MTD framework [11] is proposed that emphasizes system unpredictability through dynamic configuration variations. While effective in minimizing ransomware operational capability, this framework increases deployment complexity and lacks automation in making optimal MTD decisions. To address the insufficiency of conventional ransomware defenses that either restore encrypted data or react after infection, a file-system-level MTD platform called MTFS [12] is developed, which is built on the Linux Virtual File System using FUSE to enable full control over file operations. MTD techniques such as file access disruption, directory traversal trapping, and file-type hiding are implemented for mitigating ransomware behaviour. However, it depends on static deception mechanisms and does not have intelligent adaptation, making it less efficient against highly adaptive or stealthy ransomware.

2.2 Machine Learning-Based Moving Target Defense Approaches

Choosing suitable MTD strategies for heterogeneous and zero-day attacks remains an open research challenge, especially in resource-constrained environments. Hence, an online RL-based framework [13] is proposed to dynamically select appropriate MTD techniques for single-board computers (SBCs). It models system behaviour using behavioural fingerprinting and leverages RL for learning optimal MTD actions against different malware, such as rootkits, command-and-control attacks, and ransomware. Despite its practicality, it is inadequate for highly

harmful rootkits and does not explore scalability across several SBCs. Network Intrusion Detection Systems (NIDS) increasingly depend on ML and deep learning (DL) models, yet these systems are susceptible to adversarial examples. For this, MTD is introduced as Adversarial Defence (MTD-AD) in [14] to protect anomaly-based NIDS by stochastically changing the decision boundary. It improves robustness against both standard and adaptive adversaries by exploiting the sensitivity of adversarial samples near decision boundaries. However, it does not consider system-level resource overhead or deployment constraints. For detectors in power systems, an MTD-strengthened deep neural network (DNN) [15] is proposed that utilizes a pool of diverse models integrated with physics-based MTD via dynamic reactance perturbation. It enhances detection accuracy while minimizing retraining costs. Nevertheless, it necessitates system parameter perturbations, which are not always practical in highly sensitive grid environments. By using the MIL-STD-1553 protocol, the study [16] examines whether ML and DL models can infer and predict MTD behaviour in real-time systems. Supervised ML models can effectively detect MTD utilization, whereas Long Short-Term Memory (LSTM)-based models can forecast future address assignments under certain conditions.

In [17], MTD is considered as a partially observable stochastic game that focuses on system reimaging decisions. In order to reduce computational complexity, a structure-aware policy gradient RL algorithm is developed by exploiting the threshold structure of optimal policies. Even though theoretically scalable, practical deployment challenges remain. To address adversarial ML, an MTD strategy is proposed that switches among ML algorithms through a Stackelberg game model. Although effective under strong attacker assumptions, this strategy introduces switching costs. Table 1 shows the comparison of the existing MTD-related approaches.

Table 1. Comparative analysis of existing approaches employing MTD against ransomware and other attack scenarios

Author (Year) [Ref]	Objectives	MTD Method / Model Used	Attack Focus	MTD-Based Mitigation Techniques	Limitations
Von Der Assen et al. (2023) [10]	Mitigate malware in resource-constrained IoT devices	Lightweight multi-layer MTD	Multi-purpose IoT malware	Network randomization, runtime perturbation, data-layer shuffling	Lacks focus on ransomware file encryption
Liu and Chen (2023) [11]	Reduce the operational efficiency of ransomware in Windows environment	OS-level MTD	Data-theft ransomware	Dynamic configuration perturbation, identity/data variability	Increased deployment complexity
Von Der Assen et al. (2024) [12]	Proactively mitigate ransomware	MTFS using shuffling and deception via overlay FS	Crypto-ransomware targeting file traversal and encryption	Directory traversal trapping, deceptive file access, and file-type hiding	Lack of intelligent or adaptive decision-making
Celdrán et al. (2024) [13]	Learn optimal MTD against zero-day attacks on SBCs	Online RL	Ransomware, rootkits, C2 malware	Dynamic selection of MTD based on behavioural fingerprints	Limited MTD options
Shinde et al. (2025) [14]	Improve ransomware detection accuracy	Hybrid MTD with static, heuristic, and dynamic analysis	Polymorphic and metamorphic ransomware variants	Attack surface randomization, API obfuscation, behavioural analysis	Detection-oriented rather than preventive

He et al. (2025) [15]	Protect ML-based NIDS from adversarial examples	Stochastic MTD	Adversarial ML attacks	Randomized decision boundary shifting	Increased deployment complexity
He et al. (2025) [16]	Detect adversarial FDIAs in power systems	DNN pool and physics-based MTD	Adversarial FDIA	Model diversification, system parameter perturbation	Increased retraining overhead
Datar, and Dujardin (2025) [17]	Reduce MTD policy computation complexity	Policy Gradient RL	Probing & persistence attacks	Threshold-based reimaging strategies	Practical deployment is challenging

3. Preliminaries

3.1 Attack Model

The adversary is modelled as a ransomware attacker deploying different and evolving ransomware variants, such as RaaS, Crypto, Modern, and Locker types. Attacks are assumed to run on Windows-based systems and show realistic behaviours like encryption, file enumeration, registry manipulation, persistence establishment, process injection, and network communication. The attacker leverages sandbox evasion, delayed execution, polymorphism, and multi-stage workflows, such as triple or double extortion tactics. The attack model considers adaptive ransomware with no preceding knowledge of MTD deployment and comprises both known and zero-day families, aligned with the temporal and behavioural diversity represented in MLRan.

3.2 System Model

The proposed model contains protected endpoints or virtualized environments in which software execution is continuously monitored through runtime behavioural traces. System behaviour is captured via features such as API calls, registry activities, resource usage, file and directory operations, network traffic, and behavioural signatures. For estimating ransomware risk with confidence awareness, these temporally ordered features are processed by a Time-Series Transformer. Depending on deterministic thresholds, the system dynamically triggers MTD actions. The system supports continuous feedback through adaptive learning to adapt to evolving ransomware behaviours while maintaining low operational overhead.

3.3 Problem Formulation

Ransomware reflects time-evolving behaviour characterized by staged execution patterns that cannot be efficiently captured through static or snapshot-based defenses. Consider a protected system that generates a continuous stream of runtime behavioural events e_t such as file operations, API calls, process activities, registry modifications, resource utilization, and network interactions. These events are segmented into fixed-length sliding windows. $\{W_1, W_2, \dots, W_T\}$, in which each window denotes the system state over a short execution interval. Every window W_t is mapped to a hybrid feature vector $\mathrm{x}_t \in \mathbb{R}^d$ that merges embedded categorical events and normalized statistical activity features. The resultant temporal sequence $\mathrm{X} = (\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_T)$ captures both long-term ransomware progression stages and short-term malicious bursts.

The objective is to learn a temporal mapping:

$$f_\theta : \mathrm{X} \rightarrow (r_t, c_t) \qquad (1)$$

where $r_t \in [0,1]$ represents the ransomware risk score at time window t, and c_t represents the corresponding prediction confidence. The model should consistently differentiate ransomware activity from benign anomalies and remain robust to novel ransomware variants. On the confidence-aware risk estimation, a deterministic MTD triggering function is given by:

$$g(r_t, c_t) \rightarrow a_t \quad (2)$$

That maps model outputs to a discrete set of defense actions $a_t \in$ {No Action, Deception, Containment}. This design eliminates RL–based policy optimization, while ensuring predictability, explainability, and low-latency response. The system functions under two constraints:

Constraint 1: Real-time responsiveness is enforced in such a way that the end-to-end processing delay Δ_t for every window satisfies $\Delta_t \leq \Delta_{\max}$, allowing defense activation before large-scale file encryption happens.

Constraint 2: Defense stability is preserved by triggering MTD actions only when both confidence and risk exceed predefined thresholds, thereby avoiding unnecessary or oscillatory system reconfigurations.

Therefore, the problem is addressed by designing a scalable, low-latency temporal behavioural modelling framework that offers confidence-aware ransomware risk estimation and initiates adaptive MTD actions in a deterministic and operationally stable manner across heterogeneous execution environments.

4. Proposed Methodology

Runtime system behaviour is continuously monitored by the proposed system, and execution traces are segmented into fixed-length time windows. Each window is converted into hybrid dynamic features that integrate statistical activity metrics, embedded behavioural events, and resource-usage indicators. A Time-Series Transformer is utilized to model the temporal evolution of these features and estimate the confidence-aware ransomware risk score. Depending on predefined risk and confidence thresholds, a deterministic MTD trigger activates suitable defense actions like no action, deception, or containment, while ensuring an explainable and low-latency response. Post-MTD behaviour is again monitored, and newly observed traces are merged through self-supervised adaptive learning for maintaining robustness against emerging and zero-day ransomware. Figure 1 illustrates the overall block diagram of the proposed defense mechanism against ransomware.

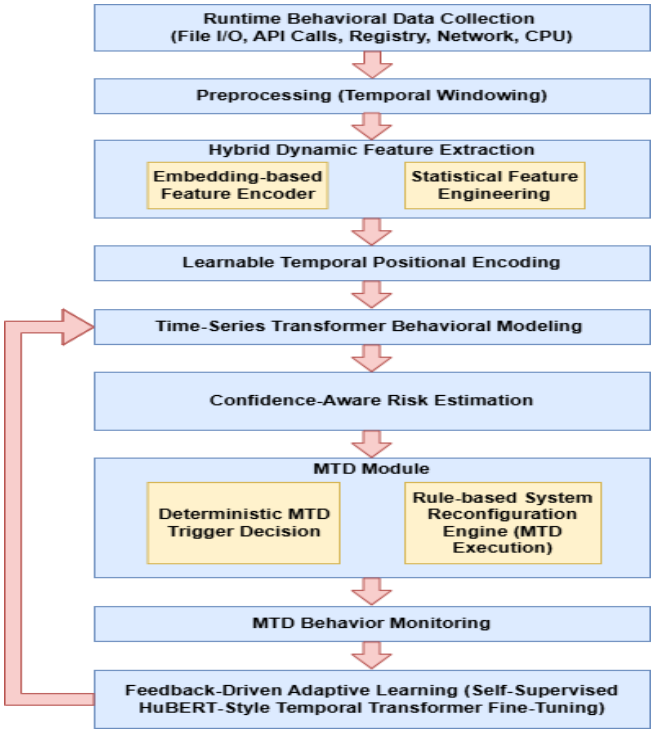

Figure 1. Overall Block Diagram

4.1 Data Collection

MLRan behavioural ransomware dataset [18] is utilized, which contains 4,880 Windows executable samples, including 2,550 goodware and 2,330 ransomware instances. Ransomware samples encompass four major

categories: RaaS, Crypto, Modern, and Locker. Each category covers 64 families gathered between 2008 and 2024, thus reflecting both historical and modern threat landscapes. Samples are obtained from several reputable sources, such as VirusTotal, VirusShare, MalwareBazaar, and curated threat intelligence repositories, while ensuring authenticity and diversity. All binaries are executed in an isolated Cuckoo Sandbox environment configured to mimic real-world environments and closely bypass sandbox-aware malware. During execution, fine-grained runtime behaviours like file and directory operations, API invocations, network communications, registry modifications, resource usage, and persistence-related activities are captured. Balanced goodware samples spanning 11 application categories are gathered and verified to ensure zero antivirus detections. It enables unbiased behavioural learning and robust discrimination between malicious and benign execution patterns.

4.2 Preprocessing

During data preprocessing, the temporal windowing module converts raw runtime behavioural traces into structured, temporally ordered sequences. System execution logs are parsed for extracting key behavioural events like process and API activities, file I/O, resource usage, registry changes, and network interactions. For preserving execution order and capturing evolving behaviour, events are aggregated within fixed-length sliding time windows (Δt). For every window, resource-usage indicators and statistical activity metrics are calculated. On the other hand, for embedding-based encoding, categorical events are indexed. Continuous features are normalized for scale consistency. To maintain uniform dimensionality, missing or sparse observations are managed through zero-padding. This streamlined preprocessing produces clean, temporally coherent, embedding-ready sequences that directly support behaviour modelling and confidence-aware MTD trigger decision and execution.

4.3 Hybrid Dynamic Feature Extraction

Hybrid dynamic feature extraction integrates categorical and numerical behavioural signals for creating rich representations of system activity. It captures both statistical patterns (resource usage, event rates) and discrete events (system operations, API calls) within temporal windows for robust ransomware modelling.

Embedding-Based Feature Encoder: Discrete categorical events are mapped to dense, learnable embedding vectors through an embedding lookup table. Formally, every event $w \in U$ is related to an embedding $\mathrm{e}_w \in \mathbb{R}^d$, which is obtained by indexing into an embedding matrix $\mathrm{E} \in \mathbb{R}^{d \times |U|}$. Events are characterized by integer indices. Embeddings within a fixed-length time window are aggregated using sum, mean, or attention-based pooling to produce a window-level semantic representation $x_t^{(c)} \in \mathbb{R}^d$. This encoding captures semantic similarity amongst related behaviors like different encryption APIs and facilitates robust generalization to novel or obfuscated ransomware variants through end-to-end learning [19].

Statistical Feature Engineering: The intensity and progression of runtime behaviours that are indicative of ransomware activity are captured. Within every sliding time window, low-level events are aggregated into meaningful statistical metrics like frequencies and rates of file operations, process creations, API calls, and network connections. Resource-usage statistics like memory, CPU, and I/O utilization are calculated to reflect abnormal execution bursts. In order to identify encryption-driven content changes, file-modification ratios and entropy-based indicators are derived. These statistics convert irregular event streams into compact, noise-resilient representations that highlight behavioural magnitude rather than individual event-level observations. When integrated with embedding-based features, the statistical descriptors offer complementary temporal cues that enhance ransomware risk estimation and support reliable, confidence-aware MTD trigger decision and execution. Statistical feature engineering from a vector $x_t^{(s)} \in \mathbb{R}^{d_s}$.

The final window-level representation is the concatenation of categorical and statistical features. This hybrid feature vector is extracted from the t-th time window:

$$x_t = \mathrm{concat}\left(x_t^{(c)}, x_t^{(s)}\right) \in \mathbb{R}^d \qquad (3)$$

This unified representation captures both discrete and continuous behavioral signals for robust ransomware modeling.

4.4 Learnable Temporal Positional Encoding

A Temporal Positional Encoding (TPE) mechanism is applied to the sequence of window-level behavioural representations for preserving execution order and evolving ransomware stages. In contrast to static sinusoidal encodings, learnable, history-aware positional embeddings are leveraged that adapt to the latest observed execution dynamics across sliding time windows [20]. A learnable positional embedding $p_t \in \mathbb{R}^d$ is generated by utilizing recent historical embeddings, which is given by:

$$p_t = F(p_{t-1}, p_{t-2}, \cdots, p_{t-H}) \quad (4)$$

where $F(\cdot)$ indicates a lightweight learnable update function. H indicates the number of historical windows considered. The TPE module captures both long-term dependencies, like reconnaissance or persistence, and short-term bursts like rapid file encryption without depending on dense attention or expensive recomputation by encoding execution sequence and temporal distance into the embedding space. These temporally enriched representations explicitly facilitate robust behavioural modelling.

4.5 Time-Series Transformer–Based Behavioral Modeling

Time-series Transformers have been extensively utilized in applications including energy demand prediction, traffic forecasting, financial trend analysis, and anomaly detection, in which modeling long-range temporal dependencies and evolving patterns is crucial. These models efficiently capture both long-term trends and short-term fluctuations through self-attention and positional encoding [21]. Inspired by this, a unified Time-Series Transformer encoder is employed for behaviour modelling and adaptive learning. During inference, it executes behavioural sequence modeling for ransomware detection. In contrast, during adaptation, the same encoder is fine-tuned using a HuBERT-style masked temporal prediction objective driven by system feedback (which is discussed in Section 4.8), enabling continuous self-supervised refinement.

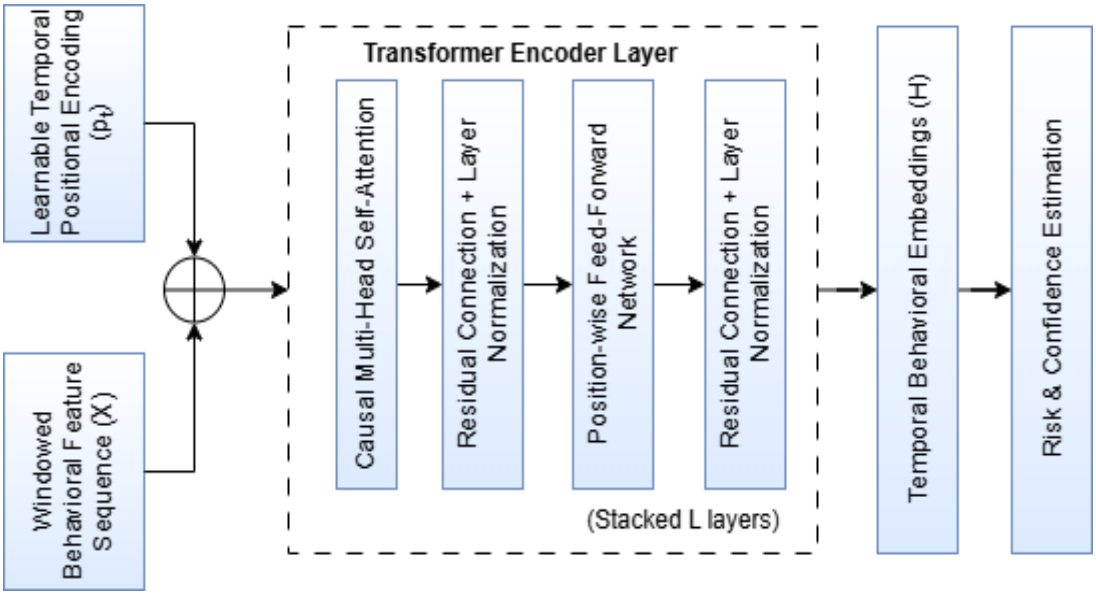

Figure 2. Time-Series Transformer–Based Behavioural Modeling Pipeline for Ransomware Behavioural Modeling

Ransomware execution unfolds through a sequence of evolving behavioural stages where malicious actions unfold gradually and are often hidden within benign system activity. The ability of the Transformer to selectively attend to critical historical events makes it appropriate for differentiating staged ransomware behaviour from normal execution. This process allows accurate and timely risk assessment for MTD activation. Capturing such behaviour necessitates a modeling framework that can learn both long-range temporal dependencies and short-term execution dynamics. Time-Series Transformer aids in modelling sequential runtime behaviour derived from windowed system execution traces.

Assume a program execution is indicated as an ordered sequence of temporal windows $\{W_1, W_2, \dots, W_T\}$, in which each window W_t collects system events observed in a fixed time interval Δt. This temporal windowing conserves execution order, minimizing dimensionality and noise in raw event streams.

Each window W_t is mapped to a hybrid feature vector $\mathrm{x}_t \in \mathbb{R}^d$ that combines embedding-based representations of categorical events and statistical activity descriptors. This hybrid encoding allows simultaneous modeling of discrete behavioural semantics and continuous execution intensity, offering a richer behavioural representation when compared to binary or count-based features. As Transformers are inherently permutation-invariant, temporal ordering is preserved by inserting learnable temporal positional encodings $\mathrm{p}_t \in \mathbb{R}^d$ (calculated in the previous section) into every window representation:

$$\mathrm{z}_t = \mathrm{x}_t + \mathrm{p}_t \tag{5}$$

The resultant input sequence $\mathrm{Z} = (\mathrm{z}_1, \mathrm{z}_2, \dots, \mathrm{z}_T)$ encodes both execution order and behavioural content, enabling the model to differentiate early-stage reconnaissance from later destructive actions.

Transformer Encoding with Causal Self-Attention: The transformer encoder with stacked multi-head self-attention and position-wise feed-forward layers processes the sequence Z. Contextual representations are calculated for each layer, which is given by:

$$\mathrm{Attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}\left(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathrm{V} \tag{6}$$

where keys K, queries Q, and values V indicate linear projections of Z. A causal attention mask is applied in such a way that the representation at time t is computed by attending exclusively to previous and present temporal windows $\{1, \dots, t\}$, thereby ensuring suitability for online detection and preventing future information leakage. Through this mechanism, the model adaptively learns which past behaviors are most related to the present execution context. This approach enables it to correlate low-intensity preparatory actions with succeeding high-impact encryption bursts, even when separated by extended temporal intervals.

At time t, the Transformer output (h_t) acts as a compact behavioral embedding that summarizes global temporal context across the entire execution history and local execution dynamics in recent windows. These behavioural embeddings directly support confidence-aware ransomware risk estimation.

4.6 Confidence-Aware Risk Estimation

The obtained behavioural embeddings are mapped to ransomware risk scores with the help of a Bayesian Neural Network (BNN) head for explicitly quantifying predictive uncertainty [22]. In contrast to deterministic classifiers, this stage models prediction confidence and ambiguity for safe MTD activation.

Let $h_t \in \mathbb{R}^d$ be the Transformer output at t and y_t be the associated ransomware risk label. Assume that y_t is generated by an unknown function $f(\cdot)$, which is parameterized by network weights W, for which a prior distribution $p(W)$is assumed. For the training dataset $\mathcal{D} = \{(h_t, y_t)\}_{t=1}^{N}$, Bayesian inference calculates the posterior:

$$p(W \mid \mathcal{D}) \propto p(\mathcal{D} \mid W)\, p(W), \tag{7}$$

which is intractable for deep neural networks. Therefore, variational inference is utilized that approximates the posterior with a tractable distribution $q(W)$ by reducing the Kullback–Leibler divergence: $\mathrm{KL}(q(W) \parallel p(W \mid \mathcal{D}))$. Monte Carlo Dropout is implemented as an effective approximation to variational Bayesian inference, in which dropout masks relate to Bernoulli random variables over network weights. During inference, T stochastic forward passes are executed to produce an approximate predictive distribution:

$$p(y_t \mid h_t) \approx \frac{1}{T}\sum_{i=1}^{T} p(y_t \mid h_t, W^{(i)}), W^{(i)} \sim q(W). \tag{8}$$

The predictive mean represents the ransomware risk score, which is given by

$$\mu_t = \mathbb{E}[y_t \mid h_t] \tag{9}$$

The predictive variance σ_t^2 captures epistemic uncertainty arising from ambiguous or previously unseen behavioural patterns. It is given by,

$$\sigma_t^2 = \mathrm{Var}[y_t \mid h_t] \tag{10}$$

This confidence-aware estimation allows risk-sensitive MTD triggering, while ensuring that aggressive defense actions are performed only when both certainty and risk are high, thus minimizing false positives, avoiding unnecessary disruption of benign system operation, and supporting explainable security decisions.

4.7 MTD Trigger Decision and Execution

The BNN produces, at each time step t, a predictive risk distribution characterized by a mean ransomware risk score. μ_t and epistemic uncertainty σ_t^2. Rather than directly coupling this output to an automated learning policy, a deterministic confidence-aware decision logic is adopted for ensuring interpretability, stability, and deployment safety.

A dual-threshold rule supervises MTD activation:

$$MTD_t = \begin{cases} 1, & \text{if } \mu_t \geq \tau_r \wedge \sigma_t^2 \leq \tau_u \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Where τ_u is the maximum tolerable predictive uncertainty and τ_r is the minimum acceptable ransomware risk level. It ensures that MTD is triggered only when malicious behaviour[23] is both high-risk and confidently detected, avoiding premature reactions under noisy or ambiguous conditions. This step eliminates adversarial manipulation of learned policies, unstable feedback loops, and unpredictable system oscillations by decoupling decision logic from policy learning.

When the trigger condition is satisfied, defense actions are performed using a rule-based system reconfiguration engine, instead of a learned controller. The MTD decision module chooses a high-level defense strategy {No Action, Deception, Containment}. Each strategy is realized through predefined system-level actions like resource relocation, service migration, node isolation, configuration randomization, and so on, while ensuring predictable and verifiable defense execution. Table 2 shows the mapping of high-level MTD strategies to concrete system actions.

Table 2. Mapping of MTD Strategies to System-Level Actions

High-Level MTD Strategy	System-Level Actions
No Action	Continuous monitoring, confidence-aware risk estimation, Logging and telemetry collection
Deception	Configuration randomization, decoy services/honeypots, Fake file systems, or credentials
Containment	Node isolation, Service migration, Resource relocation, or throttling

Consider A = $\{a_1, a_2, \ldots, a_K\}$ be the set of permissible low-level MTD actions related to the selected strategy. The chosen action a_t is determined by a fixed cost-aware policy:

$$a_t = \arg\min_{a \in A} C(a), \text{ s.t. } \mathrm{MTD}_t = 1 \tag{12}$$

where $C(a)$ is the operational cost including migration overhead, resource reallocation impact, and service disruption related to action a. By decoupling strategic MTD selection from action execution, and considering MTD as an actuation and control problem instead of a learning problem, this layer assures predictable behaviour, bounded system impact, and compliance with operational constraints. It ensures auditability, repeatability, and

explainable defense outcomes, rendering them suitable for real-world ransomware mitigation. Figure 2 shows the overall MTD process.

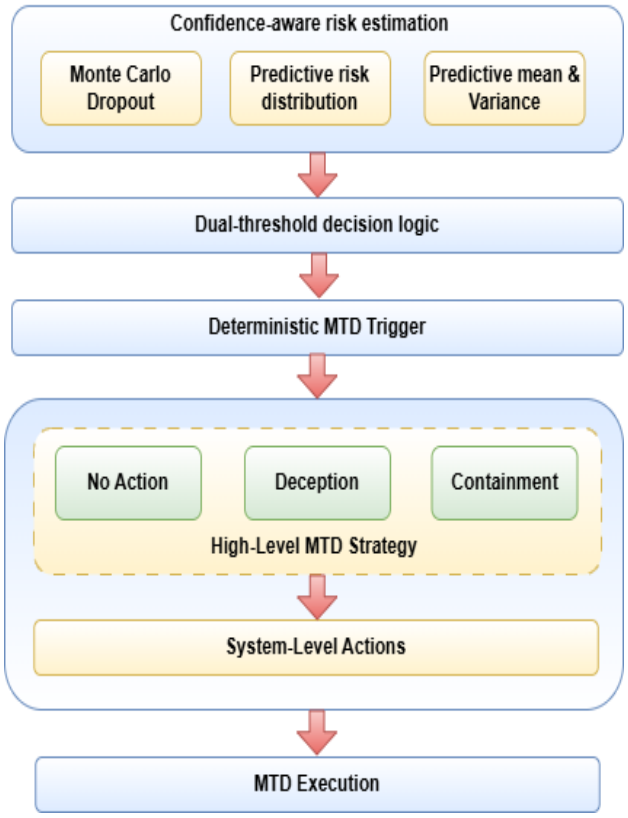

Figure 2. MTD Trigger Decision and Execution along with Confidence-aware risk estimation

4.8 Feedback-Driven Adaptive Learning

In order to provide continuous adaptation of behavioural representations under evolving ransomware strategies, self-supervised learning is utilized, which does not need labeled attack data or RL-based exploration. Time-Series Transformer encoder trained for behaviour modelling is reused and selectively fine-tuned in an adaptive mode.

MTD trigger signals and execution outcomes are fed as input to this adaptive learning phase for avoiding policy–representation leakage. Integrating defense actions directly into the learning signal may bias the behavioural encoder toward defense-induced artifacts rather than intrinsic attacker behaviour. Hence, post-MTD behavioural observations are considered for adaptive learning so that the encoder remains defense-agnostic yet defense-aware, enhancing generalization to novel ransomware strategies.

Assume $H = \{h_1, h_2, \dots, h_T\}$ be the latent behavioural sequences given as input to the transformer. $h_t \in \mathbb{R}^d$ is contextualized execution behaviour derived from file operations, system calls, execution timing, and memory access patterns. These embeddings capture how program behavior progresses under defense pressure, rendering them a suitable abstraction level for adaptive learning.

Self-Supervised Temporal Masking: For enabling adaptation without supervision, a random subset of temporal indices $\mathcal{M} \subset \{1, \dots, T\}$ is masked. This temporal masking helps in generating a corrupted sequence $\tilde{H}$. The transformer encoder processes $\tilde{H}$ and learns to recover masked execution dynamics from the surrounding temporal context.

The self-supervised objective learns to infer masked latent embeddings from the surrounding temporal context:

$$\mathcal{L}_{\text{SSL}} = \sum_{t \in \mathcal{M}} \| h_t - \hat{h}_t \|_2^2 \qquad (13)$$

where $\hat{h}_t$ indicates the predicted embedding at t.

This HuBERT-style masked temporal prediction directly functions on latent embeddings rather than raw events, eliminating relearning low-level semantics already captured during behaviour modelling. It enables the transformer to model long-range execution dependencies, sparse I/O bursts, delayed actions, and temporal irregularities—behaviors frequently leveraged by ransomware attacks for stealth and evasion.

Feedback-Triggered Adaptation: If feedback signals like behavioural deviation scores, detection confidence drift, or anomalous execution statistics represent a distributional shift, fine-tuning is activated. This selective activation avoids redundant parameter updates during benign operation, preserving stability while enabling rapid adaptation to evolving attack variants. An updated transformer parameter set is generated, which directly refines the behavioural encoder (in behaviour modelling):

$$\theta^{(1)} \leftarrow \theta^{(1)} + \eta \mathcal{L}_{\text{SSL}} \nabla_{\theta} \tag{14}$$

where η indicates the adaptive learning rate. Therefore, this process continuously improves the behavioural representations utilized for detection and MTD decision-making. This process also enables zero-day adaptation through temporal consistency learning instead of explicit labels or action-based rewards. The utilization of a unified transformer preserves low system complexity, ensuring robustness against emerging ransomware behaviors.

Algorithm:

Input: MLRan dataset D; Sliding window size Δt; Embedding matrix E; Transformer encoder parameters θ; BNN prior $p(W)$; MTD thresholds (τ_r, τ_u); MTD actions $A = \{a_1, a_2, \ldots, a_K\}$			
Output: MTD trigger decision MTD_t; executed MTD action a_t			
1.	Collect data from the MLRan dataset		
2.	for each execution trace:		
3.		Divide logs into sliding windows Δt	
4.		For each W_t:	
5.			Extract and encode events
6.			Calculate statistics
7.			Normalize features and apply zero-padding
8.		end for	
9.	end for		
	//Hybrid Dynamic Feature Extraction//		
10.	for each W_t do		
11.		for each categorical event $w \in W_t$ do	
12.			Lookup embedding $e_w \in \mathbb{R}^d$ from E
13.		end for	
14.		Aggregate embeddings → $x_t^{(c)}$	
15.		Compute window-level statistics → $x_t^{(s)}$	
16.		Concatenate $x_t^{(c)}$ and $x_t^{(s)}$ → $x_t = [x_t^{(c)}, x_t^{(s)}]$	
17.	end for		
	//Learnable Temporal Positional Encoding//		

18.	Initialize positional embeddings $p_0, \ldots, p_H$		
19.	for t = 1 to T do		
20.		Calculate p_t	
21.		Add positional embedding to the window features to form z_t	
22.	end for		
	//Time-Series Transformer Behavioural Modeling//		
23.	Input sequence $Z = \{z_1, z_2, \ldots, z_T\}$ to Transformer encoder		
24.	for each layer in Transformer do		
25.		Calculate multi-head self-attention with causal mask	
26.		Apply position-wise feed-forward layers	
27.	end for		
28.	Generate behavioural embeddings $H = \{h_1, h_2, \ldots, h_T\}$		
	//Confidence-Aware Risk Estimation//		
29.	for each h_t do		
30.		Perform T stochastic forward passes using Monte Carlo Dropout	
31.		Obtain $p(y_t \mid h_t)$	
32.		Calculate predictive mean μ_t	
33.		Calculate predictive variance σ_t^2	
34.	end for		
	//MTD Trigger Decision and Execution//		
35.	for each t do		
36.		if $\mu_t \geq \tau_r$ and $\sigma_t^2 \leq \tau_u$ then	
37.			Set $MTD_t = 1$
38.			Select high-level MTD strategy and corresponding system-level actions
39.			Execute chosen a_t
40.		else	
41.			Set $MTD_t = 0$
42.		end if	
43.	end for		
	//Feedback-Driven Adaptive Learning		
44.	while the system is running, do		
45.		if feedback signals indicate behavioural drift then	
46.			Randomly mask a subset of temporal indices $\mathcal{M} \rightarrow \tilde{H}$
47.			Input $\tilde{H}$ to Transformer
48.			Predict masked embeddings $\hat{h}_t$
49.			Compute self-supervised loss $\mathcal{L}_{SSL}$

50.			Update Transformer parameters
51.		end if	
52.	end while		

5. Results and Discussion

This section presents a comprehensive evaluation of the proposed ML-based MTD framework using the MLRan dataset. It analyses risk mitigation, detection performance, and system efficiency, emphasizing the effectiveness of adaptive defense actions against ransomware.

5.1 Dataset Description and Preprocessing

This section outlines the datasets utilized to assess the proposed ransomware detection and MTD framework and the preprocessing steps applied to ensure data quality, consistency, and suitability for efficient model training and evaluation.

Dataset Overview: For experimental evaluation, the MLRan behavioural ransomware dataset is employed. This dataset is a large-scale benchmark mainly designed for behaviour-based ransomware analysis and ML research. Instead of depending on static signatures, MLRan offers derived runtime behavioural features extracted from dynamic execution traces, allowing robust modelling of ransomware activity under realistic execution environments. The dataset is distributed in CSV format, with distinct testing and training files, namely MLRan_X_train_RFE.csv and MLRan_X_test_RFE.csv—containing behaviourally derived features with binary labels representing benign or ransomware executions.

Dataset Consolidation and Class Distribution: The original testing (976 samples) and training (3,904 samples) splits are consolidated into a unified dataset of 4,880 samples for enabling steady preprocessing, balanced resampling, global feature normalization, and flexible experimental designs. Following consolidation, the dataset includes 2,550 benign samples and 2,330 ransomware samples. Each sample is characterized using 487 discriminative behavioural features selected via mutual information filtering and recursive feature elimination, while supporting reproducibility and efficient learning.

5.2 Performance Metrics

The performance metrics considered are:

- Accuracy denotes the overall proportion of correct predictions, representing how well the model differentiates between ransomware and benign activities across all samples.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (15)$$

where TP and TN are true positives and true negatives, respectively, and FP and FN are false positives and false negatives, respectively.

- Precision is the ratio of correctly identified ransomware instances to all predicted ransomware cases, reflecting the false alarm rate and the probability of redundant MTD triggers.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (16)$$

- Recall denotes the ability of the model to correctly find actual ransomware activities, highlighting early and reliable threat identification.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (17)$$

- F1-score offers a balanced evaluation by merging precision and recall, which is particularly beneficial when handling imbalanced ransomware datasets.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (18)$$

- ROC–AUC (Receiver Operating Characteristic – Area Under Curve) calculates the ability of the model to distinguish between ransomware and benign behaviours across all possible classification thresholds. It indicates the probability that a randomly selected ransomware sample is assigned a higher score than benign samples.

$$\text{ROC-AUC} = \int_0^1 \text{TPR}\,(FPR)\; d(FPR) \tag{19}$$

where TPR and FPR indicate the true positive rate and false positive rate, respectively.

- Time-to-Detection (TTD) computes how quickly ransomware behaviour is detected after onset. This metric is essential for ransomware, in which early detection directly influences whether encryption can be successfully prevented. It measures the delay between the beginning of ransomware activity and its first successful detection:

$$\text{TTD} = t_{\text{detect}} - t_{\text{onset}} \tag{20}$$

where t_{onset} is the time at which ransomware behaviour starts, and t_{detect} is the earliest time window t so that the predicted risk score satisfies $r_t \geq \tau_r$.

- MTD Trigger Accuracy assesses the exactness of the confidence-aware trigger decision. It ensures that MTD actions are activated only when both uncertainty is low and risk is high.

$$\text{MTD}_{\text{Acc}} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\,(\,\hat{m}_t = m_t\,) \tag{21}$$

where $\hat{m}_t \in \{0,1\}$ indicates the predicted MTD trigger decision depending on thresholds (τ_r, τ_u), m_t indicates the ground-truth trigger label, and $\mathbb{I}(\cdot)$ indicates the indicator function.

- MTD Trigger Rate calculates the frequency with which MTD actions are activated during operation:

$$\text{MTD}_{\text{Rate}} = \frac{\text{Number of MTD activations}}{\text{Total time windows}} \tag{22}$$

It indicates the degree to which the framework deploys defensive reconfiguration, while balancing operational stability and responsiveness.

- Attack Success Probability Reduction (ΔASP) measures how much the MTD actions minimize the probability of a successful ransomware attack, directly reflecting the defensive efficiency of the framework. This reduction is given by:

$$\Delta\text{ASP} = P_{\text{a}}^{\text{S}} - P_{\text{a}}^{\text{MTD}} \tag{23}$$

where $P_{\text{a}}^{\text{MTD}}$ is the success probability after dynamic reconfiguration, and P_{a}^{S} is the attack success probability without MTD.

- Throughput measures the number of samples analysed in a second. In indicates the processing capacity of the detection framework, reflecting the scalability and suitability of the system for high-rate data streams in real-time environments.
- System overhead calculates the operational cost of deploying the framework, while ensuring that Transformer inference, continuous monitoring, and MTD reconfiguration remain practicable for real-time systems.

5.3 Experimental Results and Performance Evaluation

This section reports and discusses the experimental results and evaluates the performance of the proposed ML-based MTD framework. Key metrics are analysed and compared with baselines to demonstrate the robustness and real-time capabilities of the proposed framework.

5.3.2 Ransomware Detection Performance

This section provides classification performance metrics, such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC, emphasizing the model's ability to distinguish ransomware from benign executions consistently.

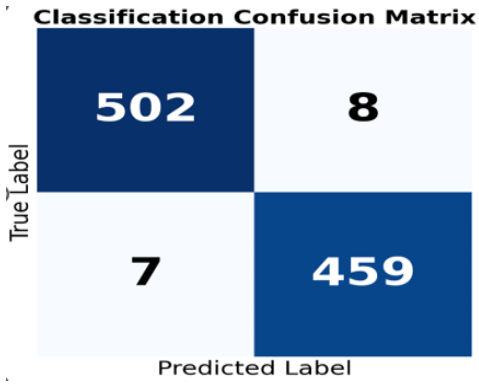

Figure 3. Confusion Matrix for Ransomware Classification

The confusion matrix in Figure 3 illustrates the classification performance of the proposed model, demonstrating the distribution of TP, TN, FP, and FN for ransomware and benign samples.

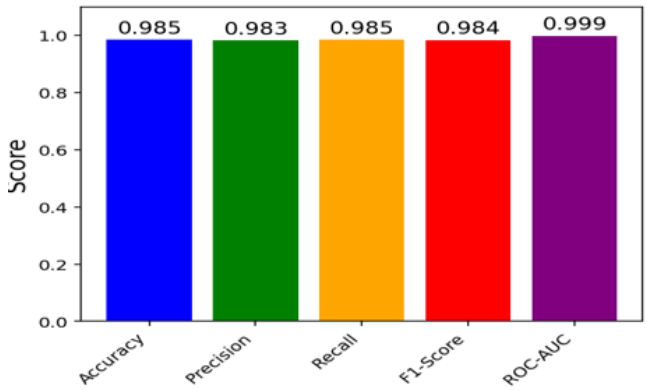

Figure 4. Overall classification performance of the proposed ML-based MTD model

Figure 4 shows the superior classification performance of the proposed model across all performance metrics. Yielding an accuracy greater than 95%, the model consistently classifies both ransomware and benign samples. High precision proves effective reduction of false positives, essential for preserving user trust and system stability, whereas strong recall ensures comprehensive ransomware threat detection. The balanced F1-score proves robustness without compromising precision or recall. Furthermore, a ROC-AUC of 0.999 highlights excellent discrimination between malicious and benign behaviours.

5.3.2.1 Comparative Analysis of Ransomware Detection Performance

The proposed framework goes beyond mere detection to proactive defense by combining uncertainty-aware risk estimation with time-series Transformer modelling. Its confidence-guided MTD triggering mechanism guarantees that defensive actions are selectively executed, minimizing false alarms and redundant system disruptions. Additionally, the inclusion of feedback-driven self-supervised adaptation enables the model to continuously fine-tune its

Table 3. Performance comparison of the proposed framework with the state-of-the-art ransomware detection models on the MLRan dataset

Methods	Models	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC / ROC-AUC
Baseline	Naïve Bayes	95.63	92.16	99.28	95.59	0.9610
Baseline	Decision Tree	94.33	92.42	95.99	94.18	0.9857

Baseline	LSTM	95.36	93.52	97.00	95.22	0.9826
Baseline	Simple Transformer	96.24	94.85	97.42	96.12	0.9850
Proposed Framework	**Time-series Transformer**	**98.46**	**98.29**	**98.50**	**98.39**	**0.9988**

understanding of ransomware behaviour over time. This integration of temporal modelling, uncertainty awareness, and adaptive defense makes the proposed framework more robust and operationally reliable for real-time ransomware mitigation when compared to the existing solutions.

5.4 MTD Trigger Effectiveness and Decision Reliability

This section evaluates the efficiency of the MTD trigger mechanism, showing how uncertainty-guided decisions minimize redundant defense actions, maintaining strong protection.

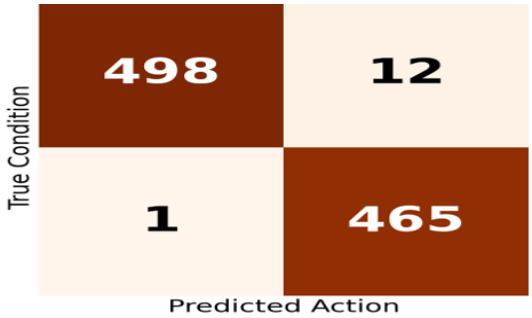

Figure 5. Confusion Matrix for MTD Trigger Decision

The confusion matrix in Figure 5 depicts the MTD trigger mechanism's performance, emphasizing correct and incorrect trigger decisions under high-confidence ransomware detection circumstances.

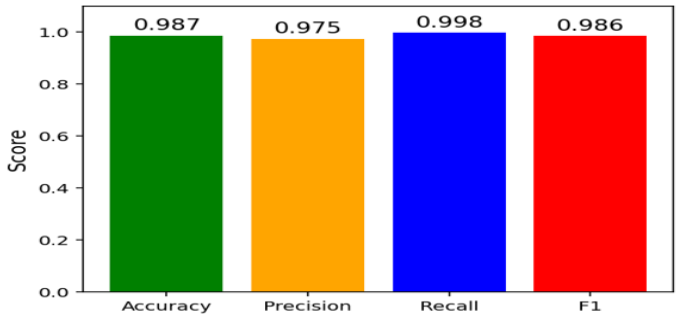

Figure 6. Performance of the MTD trigger mechanism

In Figure 6, the performance analysis of the MTD trigger mechanism exhibits high precision and accuracy, ensuring that defensive actions are initiated consistently and with reduced false alarms.

6. Conclusions

In this paper, an ML–driven MTD framework is proposed for ransomware mitigation that utilizes Transformer-based temporal behavioural modelling and confidence-aware risk estimation. The proposed framework continuously examines runtime system behaviours and captures the sequential dynamics of ransomware activity. Moreover, it enables early and reliable detection while triggering deterministic MTD actions only when both risk and prediction confidence surpass predefined thresholds. This design avoids the instability and latency related to RL–based policies, while ensuring low-latency and operationally stable defense responses. The experimental

results prove that capturing long-range temporal dependencies significantly improves precision–recall balance, detection accuracy, and early ransomware identification, effectively minimizing false alarms. Comprehensive assessment on the MLRan dataset proves substantial drops in improved system availability, attack success probability, and reliably low computational overhead, validating the real-time applicability of the framework. As the evaluation is limited to host-level behavioural analysis under controlled settings, future work will extend the framework to distributed and enterprise environments by integrating cross-host intelligence, network-level correlations, and formal policy verification for further improving robustness and operational resilience.

References

[1] Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B. C. M., & Assi, C. (2023), The Age of Ransomware: A survey on the evolution, taxonomy, and research directions. IEEE Access, 11, 40698–40723. https://doi.org/10.1109/access.2023.3268535

[2] Lee S., Kim, H. K., & Kim K. (2019), Ransomware protection using the moving target defense perspective. Computers & Electrical Engineering, 78, 288–299. https://doi.org/10.1016/j.compeleceng.2019.07.014

[3] Cen M., Jiang F., Qin X., Jiang Q., & Doss, R. (2023), Ransomware early detection: A survey. Computer Networks, 239, 110138. https://doi.org/10.1016/j.comnet.2023.110138

[4] Jalowski L, Zmuda M., & Rawski M. (2022), A Survey on Moving Target Defense for Networks: A Practical View. Electronics, 11(18), 2886. https://doi.org/10.3390/electronics11182886

[5] Zheng J., & Namin A. S. (2019), A survey on the moving Target Defense Strategies: An Architectural Perspective. Journal of Computer Science and Technology, 34(1), 207–233. https://doi.org/10.1007/s11390-019-1906-z

[6] Navas R. E., Toutain L., & Papadopoulos G. Z. (2022), Moving Target Defense Techniques for the IoT. Intelligent Security Management and Control in the IoT, 267–292. https://doi.org/10.1002/9781394156030.ch11

[7] Gurung D., Gurung S., & Pradhan M. P. (2023), Moving Target Defense - Challenges, Problems and Suitability for Internet of Things. 2023 4th International Conference on Computing and Communication Systems (I3CS), 9, 1–4. https://doi.org/10.1109/i3cs58314.2023.10127263

[8] Kritika E. (2024), A comprehensive literature review on ransomware detection using deep learning. Cyber Security and Applications, 3, 100078. https://doi.org/10.1016/j.csa.2024.100078

[9] Sharma P., Kapoor S., & Sharma R. (2022), Ransomware detection, prevention and protection in IoT devices using ML techniques based on dynamic analysis approach. International Journal of Systems Assurance Engineering and Management, 14(1), 287–296. https://doi.org/10.1007/s13198-022-01793-0

[10] Von Der Assen J., Celdrán A. H., Sánchez P. M. S., Cedeño J., Bovet G., Pérez G. M., & Stille B. (2023), A Lightweight Moving Target Defense Framework for Multi-purpose Malware Affecting IoT Devices. ICC 2023 - IEEE International Conference on Communications, 6010–6015. https://doi.org/10.1109/icc45041.2023.10278951

[11] Liu S., & Chen X. (2023), Applying moving target defense against data theft ransomware on Windows OS. Preprints.org. https://doi.org/10.20944/preprints202312.0948.v1

[12] Von Der Assen J., Celdrán A. H., Sefa R., Stiller, B., & Bovet G. (2024), MTFS: a Moving Target Defense-Enabled File System for Malware Mitigation. 2024 IEEE 49th Conference on Local Computer Networks (LCN), 1–6. https://doi.org/10.1109/lcn60385.2024.10639803

[13] Celdrán A. H., Sánchez P. M. S., Von Der Assen J., Schenk T., Bovet G., Pérez G. M., & Stiller B. (2024), RL and Fingerprinting to Select Moving Target Defense Mechanisms for Zero-Day Attacks in IoT. IEEE Transactions on Information Forensics and Security, 19, 5520–5529. https://doi.org/10.1109/tifs.2024.3402055

[14] He K., Kim D. D., & Asghar M. R. (2025), MTD-AD: Moving target defense as adversarial defense. IEEE Transactions on Dependable and Secure Computing, 22(5), 5047–5059. https://doi.org/10.1109/tdsc.2025.3560246

[15] Chen Y., Lakshminarayana S., & Poor H. V. (2025), Moving target defense against adversarial false data injection attacks in power grids. IEEE Internet of Things Journal, 12(14), 26315–26327. https://doi.org/10.1109/jiot.2025.3560127

[16] Mani G., Haliem M., Bhargava B., Manickam I., Kochpatcharin K., Kim M., Vugrin E., Wang W., Jenkins C., Angin P., & Yu M. (2023), Machine learning based resilience testing of an address randomization cyber defense. IEEE Transactions on Dependable and Secure Computing, 20(6), 4853–4867. https://doi.org/10.1109/tdsc.2023.3234561

[17] Datar M., & Dujardin Y. (2025), Adaptive learning for moving target defence: Enhancing cybersecurity strategies. arXiv.

[18] Onwuegbuche F. C., Olaoluwad A., Jurcuta A. D., & Pasquale L. (2025), MLRan: A behavioural dataset for ransomware analysis and detection. arXiv.

[19] Tokunaga H., Nicholls J., Vazhenina D., & Kanemura A. (2024), Pixel embedding: Fully quantized convolutional neural network with differentiable lookup table. arXiv.

[20] Tieu K., Fu D., Li Z., Maciejewski R., & He J. (2025), Learnable spatial-temporal positional encoding for link prediction. In Proceedings of the 42nd International Conference on Machine Learning (ICML 2025).

[21] Li S. Jin, X. Xuan Y., Zhou X., Chen W., Wang Y., & Yan X. (2019), Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. arXiv (Cornell University), 32, 5243–5253. https://arxiv.org/pdf/1907.00235

[22] Gal Y., & Ghahramani Z. (2016), Bayesian convolutional neural networks with Bernoulli approximate variational inference. In Proceedings of the International Conference on Learning Representations (ICLR).

[23] Shinde S. S., Ghoparkar S., Patil R. K., & Patil S. B. (2025), Enhancing ransomware protection through moving target defense technique. Cureus Journal of Computer Science. https://doi.org/10.7759/s44389-025-03546-z