AI-based Intent-Driven Secure System Designer

Sian En Ooi, Razvan Beuran, Yasuo Tan
Japan Advanced Institute of Science and Technology
Nomi-shi, Japan
sianen.ooi@jaist.ac.jp

Ryosuke Hotchi, Takayuki Kuroda, Yutaka Yakuwa, Takuya Kuwahara, Kozo Satoda, Norihito Fujita
NEC Corporation
Minato-ku, Japan

***Abstract*—In this paper, we present the AI-based SecureWeaver, a system designed to enhance the generation of secure network architectures. Our approach utilizes an intent-based representation of network service requirements, annotated with security constraints, and employs Design Space Exploration (DSE) augmented with classical machine learning algorithms to optimize system design. The primary input to our system consists of network service requirements and security constraints, while the output is a secure and functional network topology. By integrating AI into the existing SecureWeaver framework, our primary contribution lies in demonstrating significant performance improvements through the application of five well-established AI algorithms. Our experimental results show that the AI-based SecureWeaver achieves substantial reductions in the number of iterations and the time required to design secure systems compared to the non-AI baseline. These improvements are validated across various corporate network cases, highlighting the practical benefits of our approach. This work provides insights into the effectiveness of AI algorithms in accelerating secure design.**

***Index Terms*—Networked Systems, Automated System Design, Secure System Design, Artificial Intelligence**

I. INTRODUCTION

Modern networked systems are increasingly dynamic and heterogeneous, yet administrators must still ensure that deployed architectures meet both functional and security objectives. Intent-Based Networking (IBN) offers a compelling abstraction for expressing operator goals at a high level, decoupling what the system should achieve from how it is implemented. IBN has been applied across SDN, fog/edge platforms, and enterprise environments [2], [7], [10], and recent advances in large language models (LLMs) have expanded the ability to parse and decompose natural-language intents into actionable policies and configuration fragments [3], [4]. At the same time, standardization efforts (e.g., ETSI ZSM/ENI, 3GPP, IETF) are converging on structured representations and lifecycle stages (translation, verification, and enforcement) that are essential for automation in production settings [1], [5], [6].

Bridging the strengths of both formal and generative approaches remains necessary. LLM-driven pipelines offer broad language understanding and rapid adaptation to diverse intents, while formal-methods yield strong correctness guarantees and verifiable security properties but can be computationally costly and less adaptable to ambiguous, high-level specifications. This paper therefore positions the research as complementary to LLM-based work, it emphasizes verified-by-construction secure design and shows how AI techniques can accelerate formal search rather than replace formal verification.

Previous research introduced the AI-based Weaver, a formal-methods system designer that constructs and verifies candidate topologies using symbolic constraint solving and automated theorem proving [8]. SecureWeaver extended this approach by embedding security requirements into the specification and verification phases [9]. This paper integrates these lines of research into an AI-accelerated secure system design toolchain, AI-based SecureWeaver that accepts intent-like specifications annotated with security goals, uses AI techniques to guide and prune the search, and applies formal verification to ensure correctness and security by construction.

This paper makes the following contributions:

- Integration of SecureWeaver with AI-based Weaver: present an AI-based secure system designer that uses as input an intent-based representation of the service requirements annotated with security requirements.
- Empirical evaluation: evaluated several AI algorithms from the point of view of the performance improvements of the DSE approach in generating secure system designs.

II. AI-BASED SECUREWEAVER

This paper presents AI-based SecureWeaver, an AI-driven system that synthesizes network topologies meeting security, functional, and performance requirements. This section covers three key areas: core design framework, reinforcement learning framework, and security validation integration.

A. Core Architecture

The foundation of AI-based SecureWeaver is the AI-based Weaver, which leverages Graph Neural Networks (GNNs) to enhance design-space exploration (DSE). By modeling network topologies as graphs, the GNN rapidly evaluates candidate topologies, eliminating suboptimal options and prioritizing viable solutions. This reduces computational overheads in exploring large search spaces. The overall architecture of the AI-based Weaver system designer is shown in Fig. 1

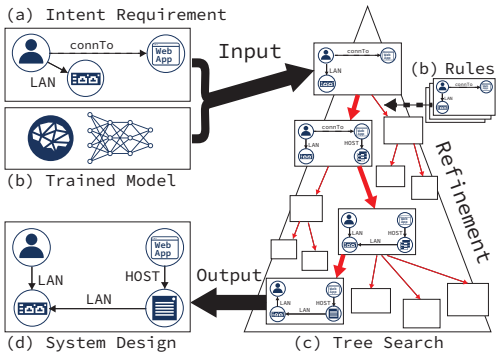

Fig. 1. Overall architecture of the AI-based Weaver system designer.

B. Reinforcement Learning Framework

AI-based Weaver integrates reinforcement learning (RL) to optimize topologies. Key algorithms include: (i) Q-learning which balances exploration (ε-greedy) with reward maximization; (ii) N-step TD and H-N-step TD (Heuristic N-step TD) which improves learning efficiency via multi-step updates and heuristic-driven ε adjustments; (iii) MCTS and MH-MCTS (Multi-Heuristic MCTS) which prioritize high-quality candidates through tree-based search and neural/gradient-guided heuristics. These algorithms collectively refine topologies while respecting system constraints.

C. SecureWeaver Integration

AI-based SecureWeaver extends the framework by embedding security verification into the design process. Each generated candidate is evaluated by the function `is_goal()`, enforcing two criteria:

1) Concreteness: The candidate must fully specify all components and relationships without any abstract entities
2) Security: The candidate must satisfy all security requirements using SecureWeaver's verification mechanism

Only configurations passing both checks receive positive rewards in the RL framework. By incorporating SecureWeaver's security verification into the reward function, after concreteness is ensured, the system guarantees that only topologies meeting qualitative, quantitative, and security requirements are accepted (see [9]). This integration ensures output topologies satisfy all required constraints.

III. Evaluation Framework for AI-based SecureWeaver

This section presents the evaluation framework of the AI-based SecureWeaver implementation. The system was developed in Python 3 and executed on an AWS EC2 p2.xlarge instance (4 vCPUs, 64 GB RAM, NVIDIA K80 GPU).

A. Evaluation Scenarios

We evaluated the AI-based SecureWeaver against three network design scenarios representing increasing complexity:

1) Case A (Baseline): Represents a corporate network with thin clients and web services vulnerable to MITRE ATT&CK threats (T1090/T1190). This scenario builds upon the deterministic evaluation from the research presented in [9].

2) Case B (Alternative Design): A contrasting configuration featuring configuration, mail, and monitoring systems with distinct threat profiles (T1602, T1190).

3) Case C (Complexity Test): This case creates a larger, more complex design scenario by combining the components and requirements from Cases A and B. The full network architecture for all cases is illustrated in Fig. 2.

B. Algorithm Evaluation Methodology

Three experiments characterized the evaluation process:

1) Algorithm Comparison (Exp #1): Baseline performance assessment of five learning algorithms with fixed hyperparameters.
2) Parameter Optimization (Exp #2): Refinement of top-performing algorithms using varying discount factors (γ) and exploration rates (ε).
3) Case Validation (Exp #3): Final testing of optimized algorithms on the complex system design requirements of Cases A-C.

To address algorithmic randomness, models were trained across multiple batches with periodic model saving (every 500 episodes). This approach enabled statistical analysis through repeated training and design cycles for both deterministic and AI-based implementations. The complete evaluation framework systematically compares AI-driven strategies against the deterministic SecureWeaver, focusing on performance stability across varying network configurations.

C. Preliminary Experiments: Deterministic versus non-deterministic DSE

The deterministic SecureWeaver (v0.1.3) averaged 214152 iterations, while the non-deterministic version (v0.1.4) averaged 397964.8 iterations (175523 to 701859). The non-deterministic approach reduced iterations by 18% in optimal cases but required up to 2.3× more iterations otherwise. Runtime analysis showed deterministic execution at ∼5500 seconds, while the non-deterministic version ranged from 3988 to 18020 seconds, averaging a 77.7% increase but occasionally reducing runtime by 26.8%. These results indicate that non-deterministic DSE introduces variability but can improve efficiency by escaping local optima.

D. Exp #1: Initial evaluation of all the five AI algorithms

We evaluated five AI algorithms to identify the most effective for secure system design. Models were saved every 500 training episodes and applied to Case A. Fig. 3 shows design iterations per episodes for each algorithm.

Algorithms were ranked based on two criteria: standard deviation below 100 and maximum iterations of 10000. If an algorithm met both conditions for a specific episode number, it was considered stable and effective.

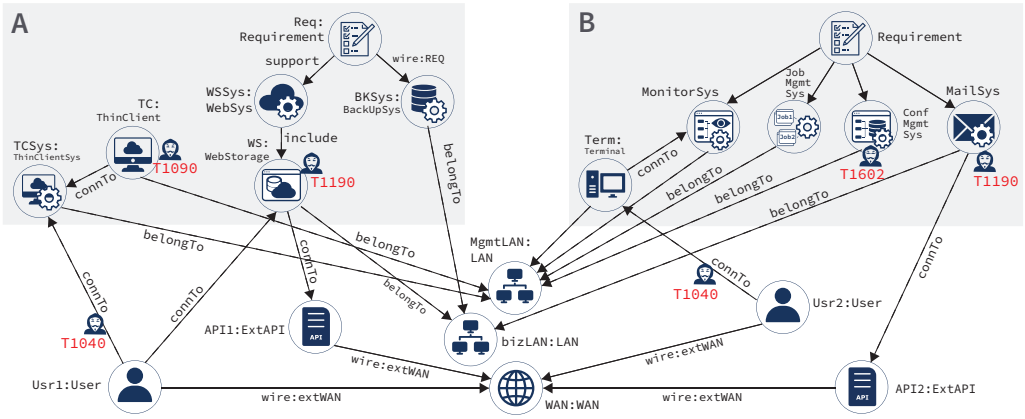

Fig. 2. Input service requirement of Case C (combination of Cases A and B).

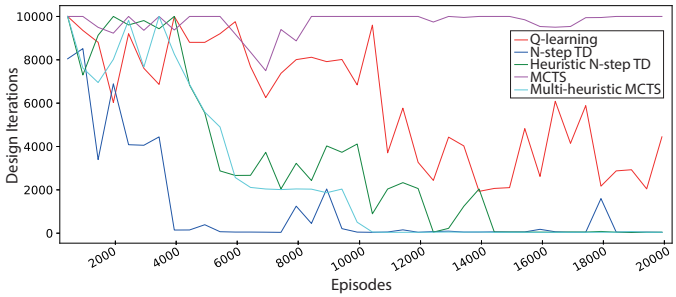

Fig. 3. The number of design iterations versus episodes for each AI algorithm in Exp #1.

N-step TD led with stable performance at 5500 episodes (*min*=34, *max*=149, μ=68.56, σ=42.6), followed by MH-MCTS at 10500 episodes (*min*=36, *max*=98, μ=53.68, σ=19.29) and H-N-step TD at 12500 episodes (*min*=34, *max*=66, μ=47.56, σ=10.95). Q-learning and MCTS did not meet these criteria and were excluded. Therefore, N-step TD, H-N-step TD, and MH-MCTS were selected for detailed evaluation in Exp #2. TD-based algorithms showed sensitivity to initial parameters, whereas MCTS, while unbiased, required more samples for stability due to higher variance.

E. Exp #2: Detailed evaluation of the best three AI algorithms

After Exp #1, we examined N-step TD, H-N-step TD, and MH-MCTS by varying γ (0.95, 0.97, 0.99) and ε-stride (0.0006188, 0.0012375). N-step TD performed best, with its top configuration at 5000 episodes (γ=0.99, ε-stride=0.0012375), achieving an average of 90 iterations (*min*=69, *max*=128, σ=17.92). Another strong setting for N-step TD was at 5500 episodes (γ=0.95, ε-stride=0.0012375), with an average of 93.36 iterations. H-N-step TD was competitive, particularly at 6000 episodes (γ=0.95, ε-stride=0.0012375), averaging 60.76 iterations (*min*=37, *max*=117, σ=30.55). MH-MCTS ranked lower, with its best configuration at 7000 episodes (γ=0.95), averaging 73.12 iterations. Results showed N-step TD as the most effective, so it was chosen for further testing in Exp #3.

TABLE I
COMPARISON OF AI AND NON-AI PERFORMANCE ACROSS CASE A AND CASE B.

Metric	Case A (Non-AI)	Case A (AI)	Δ [%]	Case B (Non-AI)	Case B (AI)	Δ [%]
Avg. Iter.	397964.8	90.0	-99.9	51830.6	39.6	-99.9
σ Iter.	204067.6	17.1	-99.9	27634.7	7.3	-99.9
Total Time [s]	9771.9	6044.2	-38.2	1004.4	14077.8	+1301.6
└ Learn Time [s]	—	6035.5	—	—	14071.5	—
└ Design Time [s]	—	8.7	—	—	6.3	—
σ Total Time [s]	5393.4	102.8	-98.1	558.4	409.0	-26.8

F. Exp #3: Additional AI-based SecureWeaver evaluation

In this set of experiments, we further evaluated the performance of the non-deterministic SecureWeaver with AI, addressing the scalability, stability, and computation efficiency.

1) Case A Results: We compared N-step TD (γ=0.99, ε-stride=0.0012375, 5000 episodes) with non-deterministic SecureWeaver (no AI). As shown in Table I, the AI approach reduced average design iterations by 99.9% (from 397964.8 to 90.0) and σ by 99.9%, showing improvements in consistency.

Average total design time decreased by 38.2% (from 9771.9 to 6044.2 seconds), with 6035.5 seconds for learning and only 8.7 seconds for execution. These results highlight the trade-off between training time and design efficiency, showcasing AI's ability to accelerate the design process significantly.

2) Case B Results: In Case B, N-step TD (γ=0.97, ε-stride=0.0012375, 10500 episodes) was compared it with non-AI SecureWeaver. AI reduced average design by 99.9% (from 51830.6 to 39.6) and σ from 27634.7 to 7.3, indicating more consistent outcomes. Total time increased by 1301.6% (from 1004.4 to 14077.8 seconds), with 99.96% spent on training (14071.5 seconds) and only 6.3 seconds on design. This highlights AI's ability to front-load computation during training, enabling rapid and reliable design generation at deployment.

3) Case C Results: Lastly, Case C evaluated the AI-based SecureWeaver on a significantly more complex system. Using

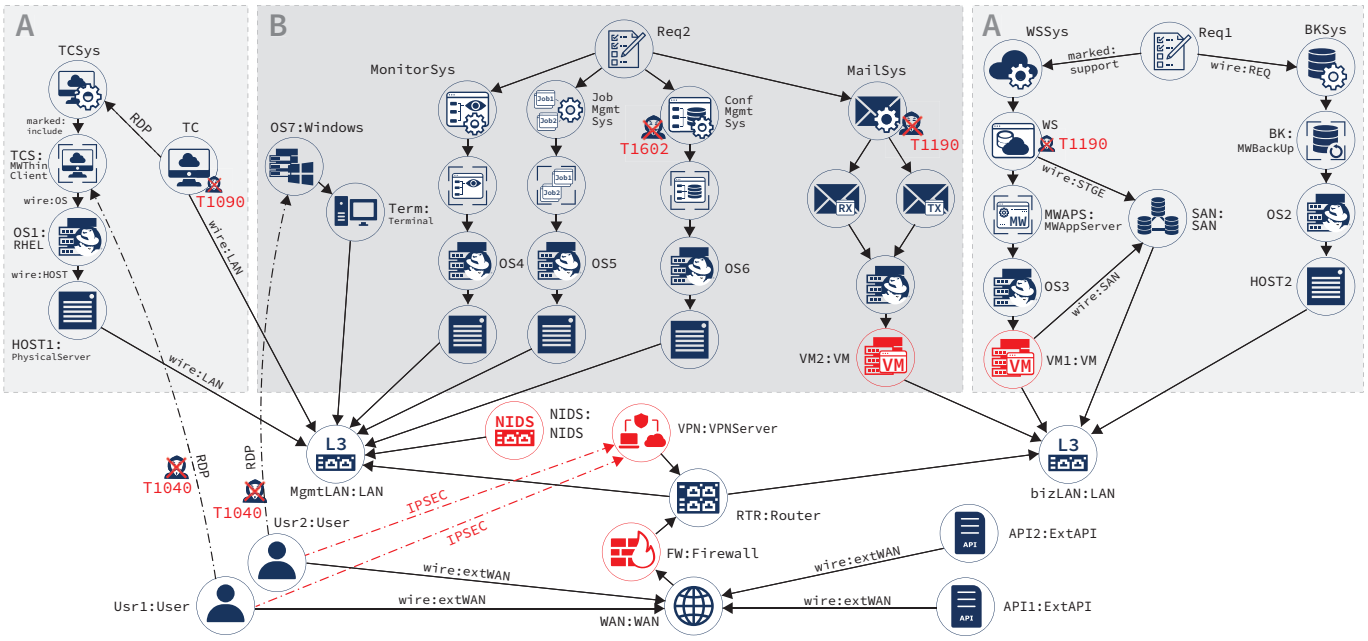

Fig. 4. System design output generated by the AI-based SecureWeaver for Case C.

N-step TD (γ=0.97, ε-stride=0.0012375), we increased the total episode number to 100000. The non-AI version failed to complete the design due to storage exhaustion, while the AI-enabled SecureWeaver succeeded in certain cases.

Figure 4 shows the output for Case C, combining elements from Cases A and B. For the Case A portion (left- and right-hand side), `TCSys`, `WSSys`, `BKSys`, and their components are fully concretized, mitigating threats T1090, T1040, and T1190. For Case B (middle), `MonitorSys`, `JobMgmtSys`, `ConfMgmtSys`, `MailSys`, and their components are also fully concretized, mitigating threats T1040, T1602, and T1190.

We ran three AI training sessions, evaluating success rates across five designs. While Training 3 achieved full success at 15000 episodes, Training 2 performed poorly across all episodes. Neither Training 1 nor 3 achieved stable convergence after 100000 episodes, though many models within these sets successfully completed all five designs. Future work will explore parameter adjustments to improve stability.

IV. CONCLUSION

This paper introduces SecureWeaver, an AI-based system designer that uses intent-based network service requirements with security annotations and employs a machine learning-enhanced DSE approach to create system designs. The updated framework incorporates security verification mechanisms from the original SecureWeaver into the new AI-based version. We evaluated the AI-based SecureWeaver through multiple experiments across various corporate network cases to identify the most suitable AI algorithms and assess the impact of AI on improving the system designer's performance. The results showed significant performance improvements, by three orders of magnitude compared to the original SecureWeaver. Future work includes exploring AI algorithm parameters to achieve more stable design results and integrating online feedback mechanisms to dynamically adapt AI algorithms during the design process. By continuously learning and adapting to changing network conditions and security constraints, the AI-based SecureWeaver could produce even more robust and secure designs.

REFERENCES

[1] 3GPP: Management and orchestration; intent driven management services for mobile networks: Ts 28.312 (2020)

[2] Collet, A., Banchs, A., Fiore, M.: Lossleap: Learning to predict for intent-based networking. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications. pp. 2138–2147. IEEE (2022)

[3] Dinh, L., Cherrared, S., Huang, X., Guillemin, F.: Towards end-to-end network intent management with large language models. arXiv preprint arXiv:2504.13589 (2025)

[4] Dzeparoska, K., Tizghadam, A., Leon-Garcia, A.: Emergence: An intent fulfillment system. IEEE Communications Magazine **62**(6), 36–41 (2024)

[5] ETSI: Experiential networked intelligence (ENI): Processing and management of intent policy. Tech. rep., ETSI (2020)

[6] ETSI: Zero-touch network and service management (ZSM): Intent-driven closed loops. Tech. rep., ETSI (2020)

[7] Jacobs, A.S., Pfitscher, R.J., Ribeiro, R.H., Ferreira, R.A., Granville, L.Z., Willinger, W., Rao, S.G.: Hey, Lumi! Using natural language for intent-based network management. In: 2021 USENIX Annual Technical Conference (USENIX ATC 21). pp. 625–639 (2021)

[8] Kuroda, T., Yakuwa, Y., Maruyama, T., Kuwahara, T., Satoda, K.: Automation of intent-based service operation with models and ai/ml. In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. pp. 1–6. IEEE (2022)

[9] Ooi, S.E., Beuran, R., Kuroda, T., Kuwahara, T., Hotchi, R., Fujita, N., Tan, Y.: Intent-driven secure system design: Methodology and implementation. Computers & Security **124**, 102955 (2023). https://doi.org/https://doi.org/10.1016/j.cose.2022.102955, https://www.sciencedirect.com/science/article/pii/S0167404822003479

[10] Sebrechts, M., Volckaert, B., De Turck, F., Yang, K., Al-Naday, M.: Fog native architecture: Intent-based workflows to take cloud native toward the edge. IEEE Communications Magazine **60**(8), 44–50 (2022)