Reasoning LLM-based Security Policy Generation for the I2NSF Framework

Quoc Pham-Nam Ho[*], Yeonwoo Park[†], Sangwon Seo[†], Damian Munoz Diaz[‡], Prasad Wasudeorao Adhau[§], Jaehoon (Paul) Jeong[†], and Tae (Tom) Oh[§]
[*]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Republic of Korea
[†]Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Republic of Korea
[‡]Department of Electrical Engineering, Ibero University, Mexico City
[§]School of Information, Rochester Institute of Technology, Rochester, NY, USA
Email: {hpnquoc, rchl, tommy2419, pauljeong}@skku.edu, a2265877@correo.uia.mx, {pa9876, thoics}@rit.edu

***Abstract*—Cloud security services rely on declarative policies for access control, but manually authoring XML-based I2NSF Consumer-Facing Interface (CFI) policies is error-prone due to complex schema constraints and contextual conditions. While Large Language Models (LLMs) can generate code, they often produce semantically incorrect or schema-violating policies. This paper proposes a closed-loop framework combining Chain-of-Thought (CoT) prompting with XSD validation to translate natural-language intents into compliant I2NSF CFI policies. This framework decomposes requests, fetches geographic context, generates draft XML via LLM, and iteratively refines it using schema feedback. A preliminary case study shows our approach reliably yields valid, constraint-complete policies where one-shot LLM generation fails.**

***Index Terms*—Large language models, I2NSF, YANG, security policy generation, XML schema, iterative refinement, verification.**

I. Introduction

Cloud security governance requires precise translation of high-level policies (e.g., temporal or geographic restrictions) into machine-enforceable formats. The IETF I2NSF framework defines the Consumer-Facing Interface (CFI) using a YANG-based data model [1]–[3] for an XML policy. It requires manual, tedious authoring and demands deep expertise, but a naive LLM-based generation approach may yield the policies that are syntactically plausible policies, but schema-invalid or semantically incomplete.

We present a schema-aware, closed-loop pipeline that generates verified I2NSF CFI policies from natural language. By integrating Chain-of-Thought (CoT) reasoning with iterative XSD validation, our system ensures outputs are both structurally compliant and semantically aligned with user intent. Contributions include: (i) a closed-loop policy generation framework; (ii) tight coupling of CoT prompting with XSD validation and self-correction; and (iii) a case study demonstrating feasibility for time- and geography-aware policies.

II. Background and Related Work

A. Reasoning with LLMs

Chain-of-Thought (CoT) prompting [4] improves LLM performance on reasoning tasks by encouraging step-by-step decomposition. It can interpret ambiguous phrases like "office hours" into precise time ranges before policy generation.

B. I2NSF and Schema Definition

The I2NSF CFI model uses YANG [5] and is serialized into XML governed by `i2nsf-cfi-policy.xsd`. This schema enforces strict hierarchy and data types, serving as the ground truth for policy validation.

C. Related Work

Existing I2NSF tools assume valid inputs and focus on CFI-to-NFI translation [6]. LLM-based configuration tools often lack rigorous schema enforcement, risking over-permissive or malformed rules. Our work bridges this gap by integrating structured reasoning with iterative, schema-driven validation.

III. Proposed Methodology

Our closed-loop pipeline (Fig. 1) comprises four phases: (1) Language Decomposition, (2) Policy Generation with CoT, (3) Schema Validation, and (4) Iterative Refinement. The LLM acts as a reasoning engine, guided by prompts that include decomposed tasks, geographic context, and prior validation errors.

A. Language Decomposition

The first step transforms the user's high-level instruction into a structured task description.

- **Intent Extraction:** Identify the core action (e.g., *block*, *allow*) and target (e.g., services, departments, regions).
- **Parameter Mapping:** Map temporal, geographic, and contextual phrases (e.g., "during office hours") to explicit fields (e.g., 09:00–17:00, Mon–Fri) aligned with I2NSF CFI attributes, using few-shot prompt templates.

B. Policy Generation with Chain-of-Thought

Once decomposed, the LLM generates the full XML policy document. We employ Chain-of-Thought prompting to guide the generation process. By forcing the model to articulate its reasoning steps before generating code, the system encourages the XML hierarchy (`<policy>` → `<rule>` → `<condition>`) to be planned logically rather than purely probabilistically.

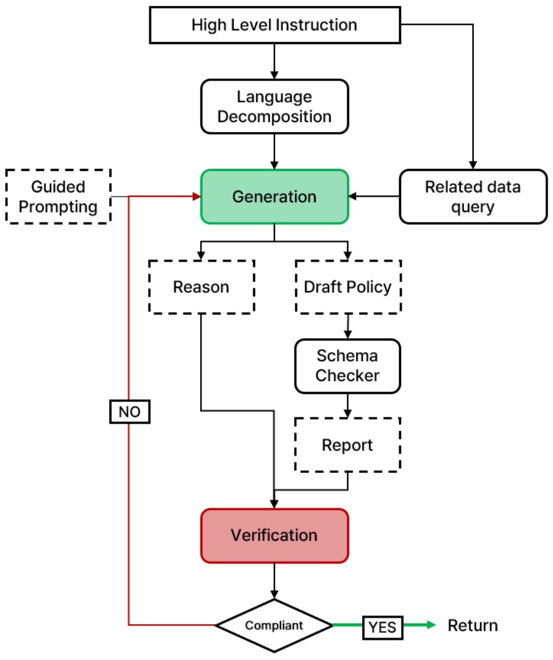

Fig. 1. Overall closed-loop pipeline for I2NSF high-level policy generation.

C. Schema Validation and Feedback Loop

The generated XML is validated against the official `i2nsf-cfi-policy.xsd` using a dedicated schema checker. This is not merely a syntax check but a rigorous structural verification. If validation fails, the engine generates a detailed error report containing the error type and location.

The system then enters a refinement loop: this error report is appended to the following prompt, instructing the LLM to self-correct the specific violation while preserving the original intent. The logical flow is formalized in Algorithm 1.

The system features a Python backend and Next.js frontend. The backend includes:

- **Policy Generator**: Orchestrates decomposition, CoT prompting, and XML assembly.
- **Region Mapping**: Uses `region.json` to resolve geographic names to I2NSF-compatible identifiers.
- **Schema Validator**: Uses `lxml` for strict XSD conformance checking.

Algorithm 1 Iterative I2NSF CFI Policy Generation and Refinement

```
Require: Natural language intent $I$, schema $S$, max iteration $N_{\max}$
Ensure: Validated XML policy $P_{\text{xml}}$ or Failure
 1: $P_{\text{tasks}} \leftarrow \textsc{Decompose}(I)$                     ▷ extract_requests
 2: $D_{\text{geo}} \leftarrow \textsc{CollectGeo}(P_{\text{tasks}})$       ▷ geo_collect (uses region.json)
 3: $(P_{\text{xml}}, E_{\text{xml}}, T_{\text{xml}}, steps) \leftarrow$ LLM_GENERATE$(P_{\text{tasks}}, D_{\text{geo}})$   ▷ generate_policy
 4: $\textsc{SaveToFile}(P_{\text{xml}})$
 5: $iter \leftarrow 0$
 6: while $iter < N_{\max}$ do
 7:     $xsd\_report \leftarrow \textsc{ValidateXSD}(P_{\text{xml}}, S)$   ▷ validate_xml_full_report
 8:     $(valid, fb) \leftarrow$ LLM_VERIFY$(P_{\text{xml}}, E_{\text{xml}}, T_{\text{xml}}, P_{\text{tasks}})$   ▷ verify_policy
 9:     if $valid$ and $xsd\_report$ has no errors then
10:         return $P_{\text{xml}}$
11:     else
12:         $(P_{\text{xml}}, E_{\text{xml}}, T_{\text{xml}}, steps) \leftarrow$ LLM_FIX$(P_{\text{xml}}, E_{\text{xml}}, T_{\text{xml}}, fb)$   ▷ fix_policy
13:         $\textsc{SaveToFile}(P_{\text{xml}})$
14:         $iter \leftarrow iter + 1$
15:     end if
16: end while
17: return Failure
```

IV. Conclusion and Future Work

We demonstrated that integrating CoT reasoning with XSD validation enables reliable generation of I2NSF CFI policies from natural language. The approach ensures schema compliance and improves interpretability via reasoning traces. Limitations include static geographic mappings and support limited to time- and location-based policies. Future work will incorporate dynamic data sources, fine-tuned open models for lower latency, and support for richer CFI features (e.g., user attributes, DPI rules).

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (No. RS-2024-00398199) and the National Research Foundation of Korea (NRF) grant (No. 2023R1A2C2002990) funded by the Korea government (MSIT). Jaehoon (Paul) Jeong is the corresponding author. GitHub Open Source: https://github.com/jaehoonpauljeong/KICS2026-Group2.

References

[1] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, and J. P. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases," RFC 8192, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8192

[2] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for Interface to Network Security Functions," RFC 8329, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8329

[3] J. P. Jeong, C. Chung, T.-J. Ahn, R. Kumar, and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-consumer-facing-interface-dm-31, May 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-i2nsf-consumer-facing-interface-dm-31

[4] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[5] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc6020

[6] P. Lingga, J. Jeong, J. Yang, and J. Kim, "SPT: Security Policy Translator for Network Security Functions in Cloud-Based Security Services," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 6, pp. 5156–5169, 2024. [Online]. Available: https://doi.org/10.1109/TDSC.2024.3371788