Mitigating Hallucination in Security Policy Generation with Large Language Models: A Prompt Ensembling Approach

Jet Wei Goh[1], Kamarul Ridwan Bin Abdul Rahim[1], Nathan Moyses[2], Vetrivel Maheswaran[3], Jaehoon (Paul) Jeong[4]
[1]Singapore University of Technology and Design, Singapore
[2]University of Alicante, Spain
[3]Rochester Institute of Technology, Rochester, NY 14623, USA
[4]Department of Computer Science & Engineering, Sungkyunkwan University, Suwon, Republic of Korea
{goh_jetwei, kamarulridwan_abdulrahim}@mymail.sutd.edu.sg, nrm69@alu.ua.es, vm6923@rit.edu, pauljeong@skku.edu

***Abstract*—Large Language Models (LLMs) have shown promise in translating high-level security intents into machine-readable policies for cloud-based security services. However, direct prompting of LLMs often produces hallucinated or invalid outputs, particularly when generating structured artifacts such as the Interface to Network Security Functions (I2NSF) Consumer-Facing Interface (CFI) policies. In this work, we propose a prompt ensembling approach that mitigates hallucination by decomposing policy generation into a sequence of specialized LLM prompts. Overall, our results show that schema-grounded prompt ensembling and intent filtering can substantially improve the reliability of LLM-driven security policy generation. The proposed design offers a practical path towards deployable policy assistants that translate natural-language intents into syntactically valid, standards-compliant I2NSF CFI policies with minimal manual correction.**

I. Introduction

Network security management in cloud environments requires translating high-level natural language intents into low-level configurations that enforce concrete security policies. The IETF's Interface to Network Security Functions (I2NSF) framework addresses this by defining a Consumer-Facing Interface (CFI) YANG data model that standardizes policy specification using an Event-Condition-Action (ECA) structure, enriched with endpoint groups (e.g., users, devices, locations, URLs, voice identifiers) and threat prevention objects (e.g., threat feeds, payload content) [1].

These constructs let users express when security rules should trigger, under which conditions, on which entities, with which threat intelligence, and with what actions, in a vendor-agnostic way that can be uniformly interpreted by Network Security Functions (NSFs). However, manually writing I2NSF CFI policies in XML is labor-intensive and demands deep familiarity with the schema. To improve usability for non-experts, Rodriguez et al. [2] proposed a Security Policy Translator (SPT) that uses an LLM to map natural-language intents to I2NSF CFI policies, demonstrating feasibility but revealing limitations in hallucination control, data availability, and robustness.

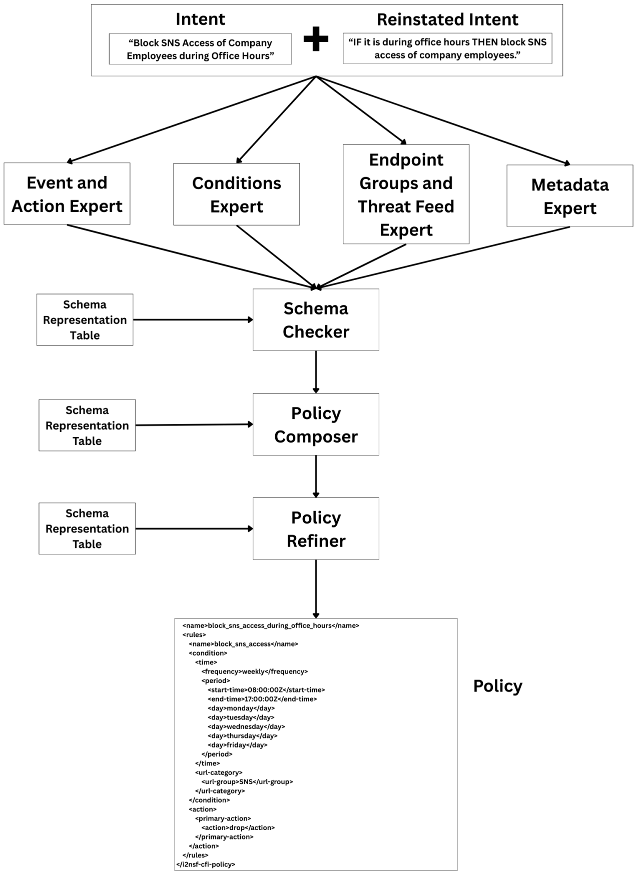
II. Problem Formulation

We study the problem of reliably generating I2NSF CFI security policies from natural-language intents using LLMs. Given a user-provided intent that describes a desired network-security behaviour, the system must (i) decide whether the intent is a valid I2NSF-style security intent and, if so, (ii) produce an XML policy that is semantically faithful to the intent and syntactically compliant with the I2NSF CFI YANG schema.

While SPT [2] shows that a single LLM call can often produce plausible policies, three key limitations remain:

1) **Schema-level hallucinations:** The LLM may invent tags, values, or structures outside the CFI YANG model (e.g., unsupported actions, wrong tag names, mis-nested `<context>` / `<firewall>` blocks, or missing endpoint groups and threat-prevention sections), causing validation failures or unintended semantics.
2) **Lack of structured training data:** There is no large, public dataset of aligned intent–policy pairs, limiting data-driven approaches such as retrieval over similar intents, supervised fine-tuning, or knowledge-graph embedding models that directly map intents into the I2NSF policy space.
3) **No intent relevance filtering:** The baseline pipeline assumes every input is a valid security intent. In realistic deployments, however, many queries are non-security related; passing these directly to the generator can yield syntactically plausible but semantically meaningless I2NSF policies (see Section III-B1).

In this paper, we address the above challenges by proposing a prompt ensembling approach [3] for security policy generation. The implementation of the prompt ensembling pipeline, together with the synthetic intent datasets and evaluation scripts, is publicly available in our project repository at https://github.com/jaehoonpauljeong/KICS2026-Group3.


Intent
"Block SNS Access of Company Employees during Office Hours"
Reinstated Intent
"IF it is during office hours THEN block SNS access of company employees."
Event and Action Expert
Conditions Expert
Endpoint Groups and Threat Feed Expert
Metadata Expert
Schema Representation Table
Schema Checker
Schema Representation Table
Policy Composer
Schema Representation Table
Policy Refiner
Policy

```xml
  <name>block_sns_access_during_office_hours</name>
  <rules>
    <name>block_sns_access</name>
    <condition>
      <time>
        <frequency>weekly</frequency>
        <period>
          <start-time>08:00:00Z</start-time>
          <end-time>17:00:00Z</end-time>
          <day>monday</day>
          <day>tuesday</day>
          <day>wednesday</day>
          <day>thursday</day>
          <day>friday</day>
        </period>
      </time>
      <url-category>
        <url-group>SNS</url-group>
      </url-category>
    </condition>
    <action>
      <primary-action>
        <action>drop</action>
      </primary-action>
    </action>
  </rules>
</i2nsf-cfi-policy>
```

Fig. 1: Architecture of the Prompt Engineering approach.

III. Methodology

A. Prompt Ensembling Pipeline

Instead of using a single prompt to generate an XML policy, we decompose the task into multiple specialized LLM prompt "experts" (e.g., for events/actions, conditions, endpoint groups, threat feeds, and metadata). Each expert produces a focused intermediate output, which is then checked against a Schema Reference Table derived from the I2NSF CFI YANG model and finally composed into a complete XML policy. This staged, schema-grounded design narrows each prompt's scope, reduces hallucinations, and helps ensure that the resulting policy is both faithful to the user's intent and syntactically compliant with the I2NSF CFI model, as illustrated in Fig. 1.

The pipeline proceeds as follows:

1) **Intent Restatement and Expert Extraction:** The natural-language intent is first normalized into an IF–THEN statement. This canonical form, together with the original intent, is passed to a small set of LLM prompt "experts" that each extract one component of the policy: (i) events and actions, (ii) contextual conditions (e.g. time, source, destination), (iii) endpoint groups and threat feeds, and (iv) policy metadata (names, language, priority, resolution strategy).
2) **Schema Readiness Checker**: Validate extracted components against a CSV-based Schema Reference Table of the I2NSF CFI YANG model, flagging non-compliant values and thereby constraining free-form outputs to the allowed schema vocabulary.
3) **Policy Composer**: Map the validated components into a full I2NSF CFI XML policy using only approved tags and structures, treating composition as deterministic field mapping rather than creative generation to avoid tag- or structure-level hallucinations.
4) **Policy Refiner**: Apply a final schema-guided refinement pass to normalize field order and structure and remove residual hallucinated details (e.g., invented IPs, MACs, URLs) before handing the policy to external tools.

B. Intent Relevance Filtering

1) Risk of Processing Irrelevant Intents: In practice, a policy assistant will receive a mix of security-related and unrelated queries (e.g., meeting bookings, financial calculations). If all inputs are blindly fed into the policy generator, the LLM may produce either (i) syntactically invalid policies that validators can catch, or (ii) syntactically valid yet semantically irrelevant policies that are more dangerous because they may be deployed and silently alter network behaviour. This motivates a dedicated intent relevance filter in front of the pipeline.

2) Intent Relevance Filtering with an LLM Classifier: To address this, we introduce a schema-grounded intent relevance classifier as a pre-filter in front of the prompt-ensembling pipeline. This classifier is implemented as an additional LLM prompt ”expert” whose only task is to decide whether a given intent is suitable for I2NSF policy generation with the Schema Reference Table. Only intents classified as valid are passed to the prompt ensembling pipeline. Intents classified as not valid are rejected early, thereby preventing the pipeline from hallucinating seemingly plausible policies from irrelevant, off-domain intents.

IV. Performance Evaluation

Our evaluation covers three aspects: (1) the syntactic correctness of generated policies, (2) a qualitative comparison of policy outputs for a representative intent, and (3) the accuracy of an LLM-based intent relevance classifier that filters off-domain inputs before policy generation. All experiments use the official I2NSF CFI YANG model as the ground truth for validation. For policy generation we compare a single-step baseline with our prompt ensembling pipeline instantiated with GPT-4o-mini and GPT-5-mini, and for intent relevance filtering we compare the GPT-4o-mini and GPT-5-mini LLM models.

A. Syntactic Correctness Comparison

We generated 50 synthetic one-sentence security intents and evaluated three methods: (i) a baseline single-step GPT-4o-mini prompt that directly outputs XML, (ii) our prompt

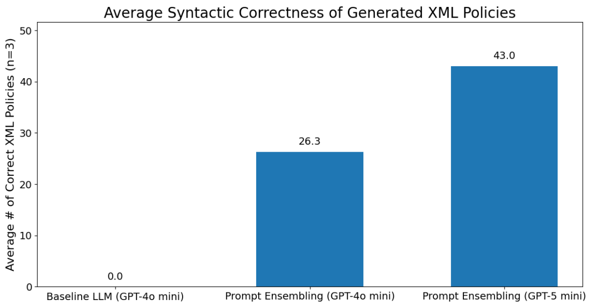


Fig. 2: Average number of syntactically correct XML policies over three runs for each method.

ensembling pipeline with GPT-4o-mini, and (iii) the same pipeline with GPT-5-mini. Each configuration was run three times over all 50 intents, and we measured how many resulting XML policies were syntactically valid according to yanglint [4] and the I2NSF CFI YANG schema.

As shown in Fig. 2, the baseline LLM never produced a schema-compliant policy. In contrast, the prompt ensembling pipeline achieved on average 52.6% syntactic correctness with GPT-4o-mini and 86% with GPT-5-mini, demonstrating substantial gains from schema-grounded ensembling and further improvements from a stronger backbone model.

B. Qualitative Output Comparison

We use the intent "Block SNS Access during Business Hours" to compare the baseline single-prompt LLM against our prompt-ensembling pipeline. Table I summarizes their compliance with key elements of the I2NSF CFI YANG schema.

TABLE I: I2NSF CFI YANG schema properties for the "Block SNS Access during Business Hours" intent.

XML Policy Aspect	Baseline LLM	Prompt Ensembling
Explicit policy metadata (language, priority, resolution)	×	✓
Endpoint groups defined and referenced	×	✓
Valid `<firewall>` node under `<condition>`	×	✓
URL category bound to URL group	×	✓
Time context	Partial	✓
Primary action value (`drop`)	✓	✓
Schema-valid	×	✓

The baseline LLM correctly infers that SNS traffic during business hours should be dropped, but produces a schema-invalid policy: it references an undefined generic source, omits URL and destination groupings, introduces a non-existent `<firewall-condition>` child under `<condition>`, and wraps the action in an `<actions>` container instead of the schema-defined `<action>`, all of which cause yanglint validation failures.

In contrast, the schema-grounded prompt ensembling pipeline (using GPT-5-mini) generates a fully valid policy. It defines and reuses endpoint groups for corporate users and SNS sites, encodes business hours as a recurring weekday context, and adds consistent metadata while preserving the intended `drop` action. Across similar intents, single-step outputs often require substantial manual repair, whereas ensembling typically yields policies that are deployment-ready or need only minor adjustments.

C. Evaluation of LLM Classifier for Intent Relevance Filtering

We evaluated the LLM intent relevance classifier on a labeled dataset of 100 synthetic policy intents (50 security-related, 50 irrelevant), and compared two backbone models: GPT-4o-mini and GPT-5-mini. For each intent, the classifier predicts if the intent is either valid or not valid.

GPT-4o-mini correctly classified 91 out of 100 intents, achieving an accuracy of 91%. GPT-5-mini correctly classified 97 out of 100 intents, achieving an accuracy of 97%. These results confirm that an LLM-based pre-filter can effectively prevent off-domain intents from entering the policy generation pipeline, thereby reducing the risk of hallucinated but syntactically valid policies being deployed.

V. Conclusion

We proposed a schema-grounded prompt ensembling method for generating I2NSF CFI security policies from natural-language intents using LLMs. By splitting policy generation into specialized experts (events/actions, conditions, endpoint groups and threat feeds, metadata) and validating their outputs against a Schema Reference Table derived from the I2NSF CFI YANG model, we significantly reduce hallucinated tags, invalid values, and structural errors compared to single-step prompting. In addition, an LLM-based intent relevance classifier filters out non-security intents before policy generation, further improving robustness and safety.

VI. Acknowledgement

The authors would like to thank the I2NSF Working Group for providing the open data model and the anonymous reviewers for their valuable feedback. Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] J. P. Jeong, C. Chung, T.-J. Ahn, R. Kumar, and S. Hares, "I2NSF consumer-facing interface YANG data model," Internet-Draft draft-ietf-i2nsf-consumer-facing-interface-dm, 2023, [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-consumer-facing-interface-dm/.

[2] M. L. Rodriguez, J. A. Berasategui, and J. P. Jeong, "Security policy generation for cloud-based security services using large language model," in *Proc. KICS Winter General Conf.*, 2025, [Online]. Available: http://iotlab.skku.edu/publications/domestic-conference/KICS-2025-Winter-LLM-Based-Security-Policy-Generation.pdf.

[3] S. Schulhoff, "Prompt ensembling," DiVeRSe and AMA for Accurate LLM Results, 2023, [Online]. Available: https://learnprompting.org/docs/reliability/ensembling.

[4] CESNET, "libyang: YANG data modeling language library," 2024, [Online]. Available: https://github.com/CESNET/libyang.