Kunerva+: An Intelligent Security Policy Generation Framework for Containers

BOM KIM, YUJUNG CHIO, SEUNGSOO LEE*

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INCHEON NATIONAL UNIVERSITY

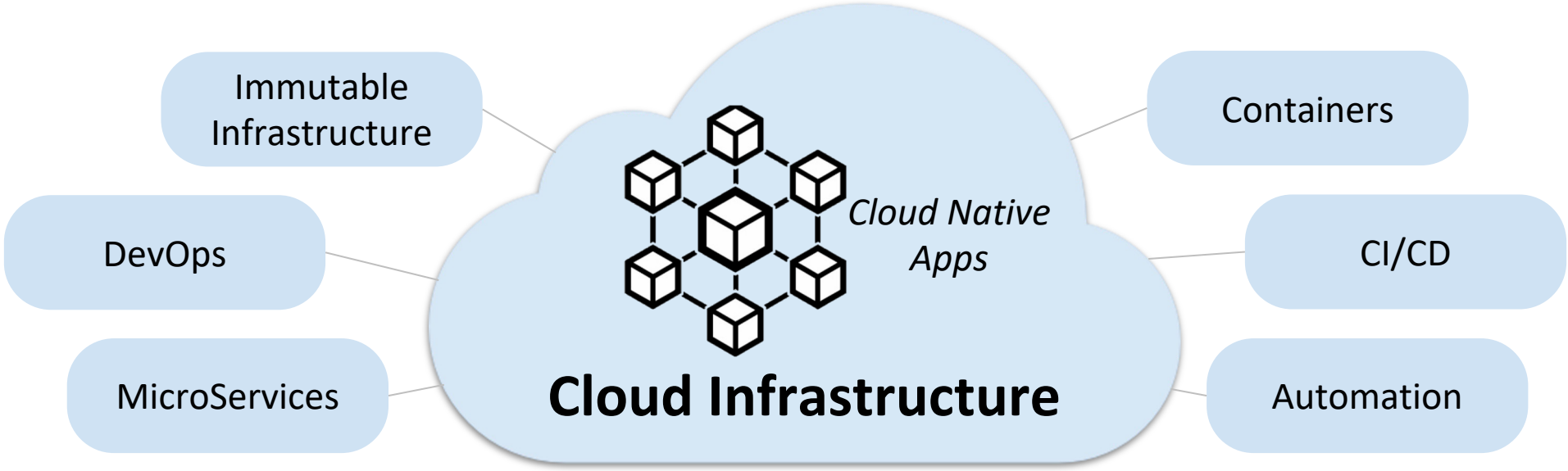
2024/06/20 @ 한국정보보호학회 하계학술대회 (CISC-S' 24)

OUTLINE

1. Problem Statements
2. Related Work
3. Kunerva+ Design
4. Evaluation - Performance, Use Case
5. Conclusion and Future Work

PROBLEM STATEMENTS

- **Increased complexity of security management**
  - Frequent Creation and Deletion of Containers
  - Dynamic Network Configuration Changes
  - Increased risk of security vulnerabilities



PROBLEM STATEMENTS

- **Limitations of Passive Security Policy Management**

Misconfiguration

Lack of response to real-time security needs

Ineffective security policy configuration

Time-consuming and error-prone manual management

Gaps between security requirements and actual policies

Requires expert-level knowledge

Difficulty in ensuring consistency and accuracy

RELATED WORK

- **Jacobs et al. (LUMI) [1]** proposed using natural language to express network management intent with Nile.
- But this approach ***utilizes an intermediate form of policy***, rather than using natural language verbatim, and ***is primarily focused on network configuration***.

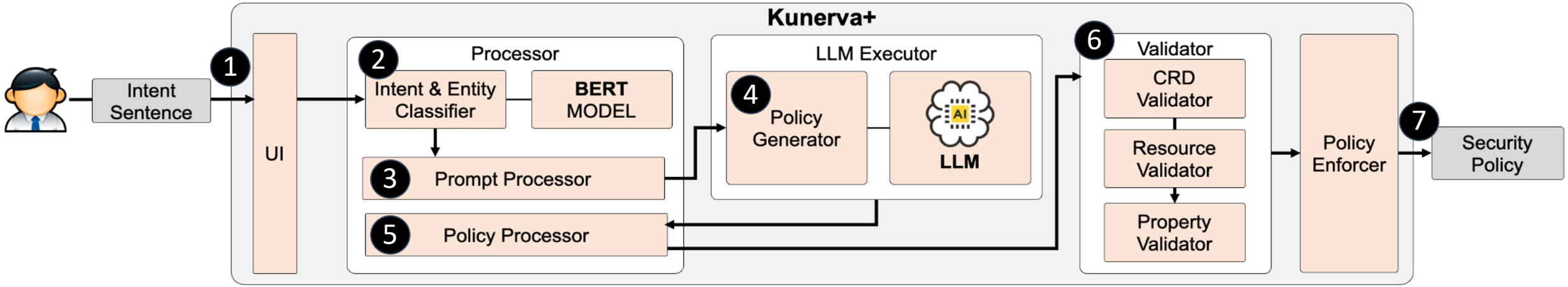
- **Li et al. (AUTOARMOR) [2]** developed an automatic policy generation method for inter-service access control in microservices.
- Although this method effectively handles static analysis and policy updates, it is ***not comprehensive enough to integrate both network and system security policies*** in cloud-native environments.

[1] Jacobs, Arthur S., et al. "Hey, lumi! using natural language for {intent-based} network management." 2021 USENIX Annual Technical Conference (USENIX ATC 21). 2021.
[2] Li, Xing, et al. "Automatic policy generation for {Inter-Service} access control of microservices."30th USENIX Security Symposium (USENIX Security 21). 2021.

Kunerva+ DESIGN: Architecture

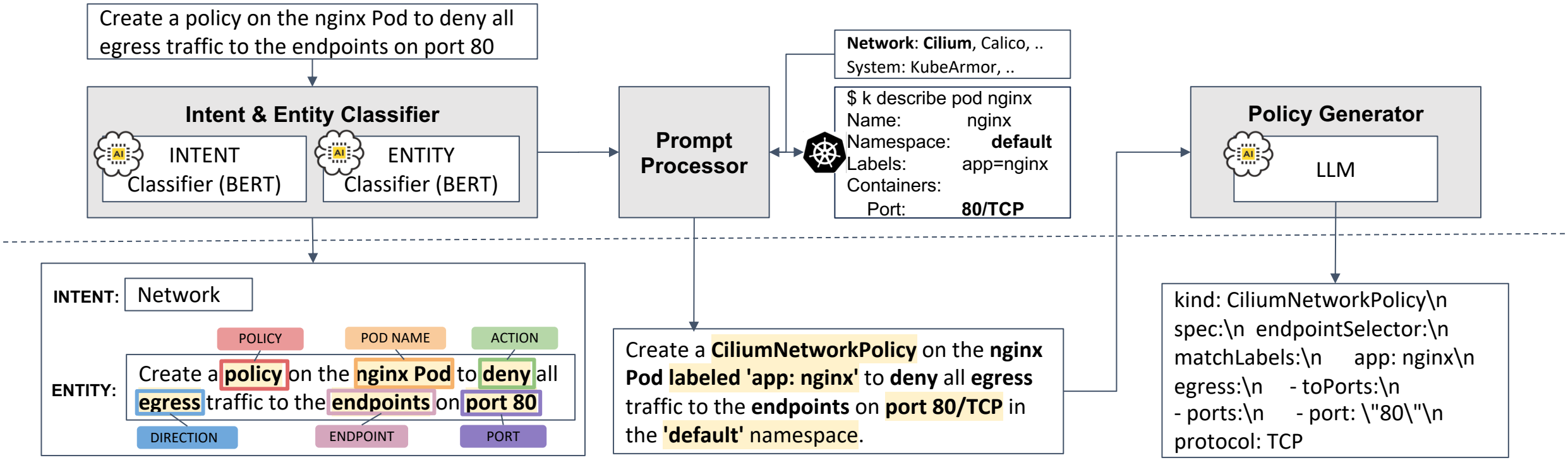
- Aim to enhance the efficiency of security management in cloud-native **by *automatically generating and validating network and system security policies*** based on natural language input.

1. <u>Accurate interpretation of natural language input</u>: Be able to accurately analyze natural language input from users and translate it into security policies.
2. <u>Validate and enforce automated policies</u>: Automate the process of validating and enforcing the generated security policies.



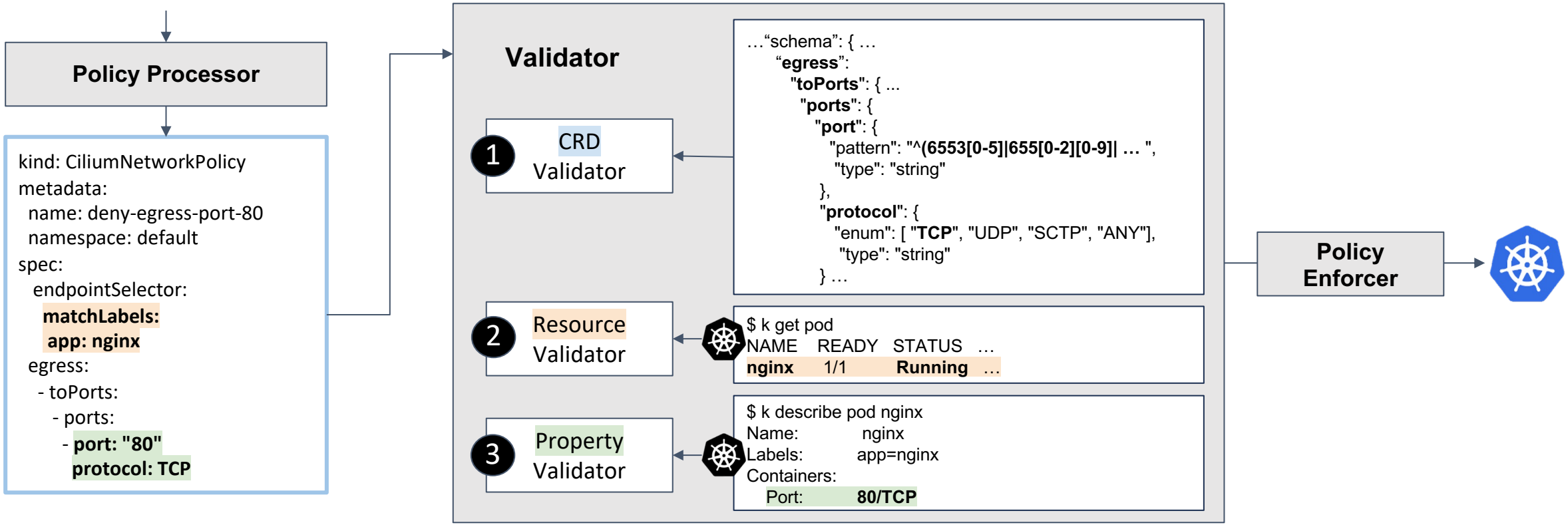
Kunerva+ DESIGN: Policy Creation

- Use BERT-based classifiers to identify intents and entities from user input.
- Transform the extracted information into detailed prompts for the policy generator.



Kunerva+ DESIGN: Policy Validation and Enforcement

- Verify CRD syntax, resource existence, and property conditions.
- Ensure that the policy is correctly configured and applicable.



Kunerva+ DESIGN: Select Dataset and Model

- Scraped policy files ***from GitHub***, modified fields, and generated instructions.
- Choose an ***open text-to-text model*** for policy creation:
  1. Models with small size and low parameter count for high accuracy
  2. Models with above-average policy performance for a wide range of generation requirements

Dataset Name	Type	Size	
Network Policy	JSON Lines	166,064개	187.3 MB
System Policy		2,914개	3.27 MB
Network Policy	CSV	1,500개	4.10 MB (3,000개)
System Policy		1,500개	

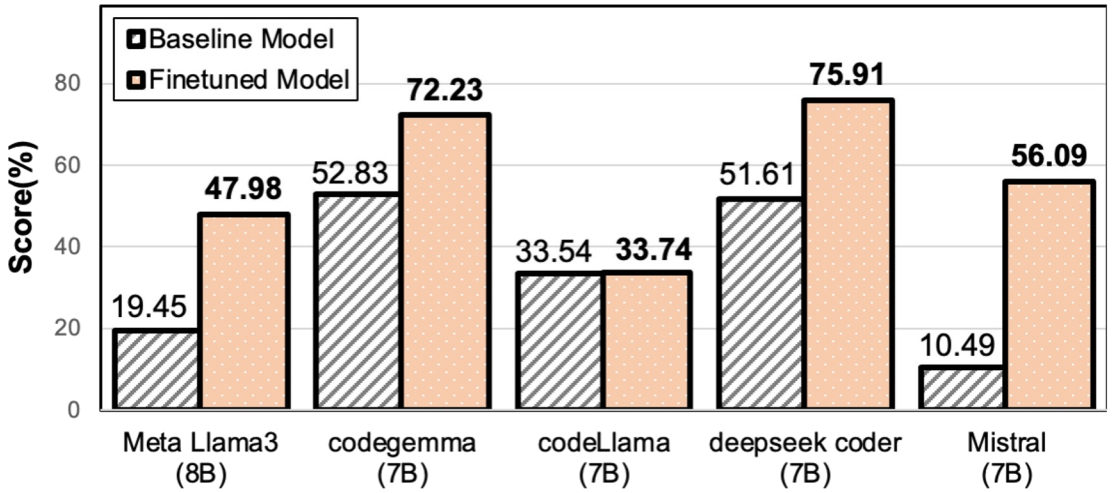
Table 1. Summary of Datasets

Model Name	Size (params)	Purpose
bert-intent-classification	425 MB	Intent Classification
bert-ner-classification	425 MB	NER Classification
Meta/Meta-Llama-3-8B-Instruct	16GB (8.03B)	Policy Generation
DeepSeek/deepseek-coder-7b-instruct-v1.5	14GB (6.91B)	Policy Generation
MistralAI/Mistral-7B-Instruct-v0.2	15GB (7.24B)	Policy Generation
Google/codegemma-7b-it	17GB (8.54B)	Policy Generation
Meta/codeLlama-7b-Instruct-hf	14GB (6.74B)	Policy Generation

Table 2. Summary of Model Features

EVALUATION: Performance of models (BLEU)

- On average, the fine-tuned model ***outperformed*** the baseline model by ***approximately 27%***.



Test Instruction	Desired Output
Create a **CiliumNetworkPolicy** allow all of **egress** traffic from endpoints **labeled with 'app: myService'** to the external IP **'10.0.10.2/32'.**	Kind: CiliumNetworkPolicy\n spec:\n endpointSelector:\n matchLabels:\napp: myService\n egress:\n - toCIDR:\n - 10.0.10.2/32

Predicted Output

Baseline Model

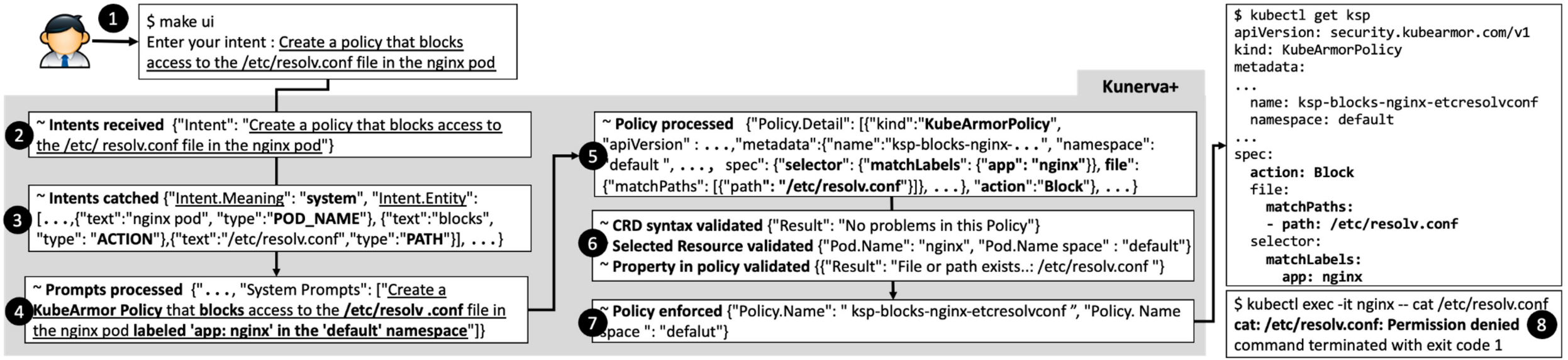
```
kind: CiliumNetworkPolicy
spec:
  endpointSelector:
    matchLabels:
      app: mySerivce
  egress:
    - toEndPoints:
      - matchLabels:
          ip: 10.0.10.2/32
```

Fine-tuned Model

```
kind: CiliumNetworkPolicy
spec:
  endpointSelector:
    matchLabels:
      app: mySerivce
  egress:
    - toCIDR:
      - 10.0.10.2/32
```

EVALUATION: Use Case

- **Example:** Blocking access to the /etc/resolv.conf file in the nginx pod.
- Demonstrated policy creation, validation, and enforcement process to show the ability of Kunerva+ to effectively manage security policies in real-world scenarios.



CONCLUSION AND FUTURE WORK

- Provide an ***intelligent security policy generation framework*** to reduce complexity and human error in containerized environments.

- Propose an ***automated policy validation and enforcement mechanism*** to ensure reliability and accuracy in dynamic cloud-native environments.

- Demonstrate the ***practical utility of AI in security management by using fine-tuned LLMs*** to effectively translate natural language input into accurate security policies.

- Future work focuses on **automatically inferring optimal security policies based solely on resource configuration files** to further reduce user input while improving the system's adaptability to security needs.

THANK YOU