AWS 환경에서 대규모 언어모델 기반 코드형 클라우드 인프라의 목표 지향적 보안 최적화 연구

유성연[o1], 김영재[1],

[1]서강대학교 AI・SW대학원,

syyoo@nongshim.co.kr, youkim@sogang.ac.kr

Goal-Oriented Security Optimization of Infrastructure as Code Based on Large Language Models in AWS Environments

[1] Seongyeon Yoo[O], Youngjae Kim,

[1] Graduate School of AI・SW, Sogang University

요 약

클라우드 인프라를 코드로 관리하는 방식이 일반화되면서, 코드에 남아 있는 보안 허점이 그대로 운영 환경에 반영되는 문제가 커지고 있다. 현재 보안 점검 도구는 코드에서 취약한 부분을 찾아줄 수 있으나, 이를 실제로 교정하는 과정에는 상당한 전문 지식과 시간이 요구된다. 본 연구는 대규모 언어모델이 보안 규칙을 참고해 취약한 인프라 코드를 자동으로 교정하는 시스템을 설계하고, 규칙 제공량과 교정 성능 사이의 관계를 체계적으로 분석하였다. 규칙이 부족하면 교정의 근거가 불충분하고, 과다하면 정보 과부하로 인해 오히려 성능이 저하된다. 이 양극단 사이에 교정 효과가 극대화되는 최적 구간이 존재함을 실험적으로 확인하였다. 제안 시스템은 상황에 적합한 규칙을 선별하고, 교정안의 생성과 검토를 반복하는 다단계 협상 구조로 동작한다. 이 효과는 실제 기업 환경의 취약 코드와 외부 벤치마크에서도 동일하게 재현되었으며, 단순한 반복 시도와는 질적으로 구별되는 구조적 이점임을 통제 실험으로 검증하였다. 본 연구는 언어모델에게 무엇을 얼마만큼 알려줄 것인가가 보안 교정의 성패를 가르는 핵심 설계 요소임을 밝히며, 교정 과정에서 개발자의 부담을 줄이고 인프라 보안 수준을 효율적으로 높일 수 있는 가능성을 제시한다.

1. 서 론

클라우드 인프라를 코드로 정의하고 자동 배포하는 코드형 인프라(Infrastructure as Code, IaC) 방식이 보편화되면서, 코드에 내재된 보안 취약점이 운영 환경에 그대로 반영되는 문제가 대두되고 있다. 이에 대규모 언어모델(LLM)을 활용하여 보안 규칙을 프롬프트에 제공하고 취약 코드를 자동 교정하려는 시도가 나타나고 있으나, 규칙을 많이 제공할수록 오히려 유효한 교정안까지 억제되는 현상이 관찰된다. 이는 학습 단계에서 발생하는 과잉 거부(over-refusal)와 달리 [2], 추론 시점에 프롬프트 내 규칙 밀도가 높아지면서 모델의 생성 자유도가 제약된다.

본 연구의 기여는 세 가지이다. 첫째, 규칙 제공 비율(RCR, Rule Coverage Ratio)에 따른 교정 성능의 구간별 변화 양상을 대규모 실험을 통해 처음으로 정량화하였다. [3] 둘째, 제안-검토-정제를 반복하는 교정 에이전트가 규칙 제공 구간에 따라 서로 다른 역할을 수행함을 증명하고자 한다. 최적 구간에서는 규칙 부족으로 실패한 교정을 복구하고, 규칙 과다 구간에서는 불필요한 생성 오류를 억제하는 행동을 동일한 실험환경 조건에서 비교하였다. 본 연구는 동일한 교정 알고리즘이 규칙 제공량에 따라 질적으로 다른 행동을 보인다는 구간 의존적 특성을 처음으로 식별한 것에 의의가 있다.

2. 연구배경

2.1. 관련연구

LLM 기반 코드 생성의 평가는 HumanEval[7]이 제안한 Pass@k를 기점으로 체계화되었다. Pass@k는 모델이 k번 시도했을 때 적어도 하나의 정답 코드를 생성할 확률로, 특히 Pass@1(단일 시도 성공률)은 코드 생성 능력의 사실상 표준 지표로 자리 잡았다. 이후 단순 통과 여부를 넘어 지시 조건을 얼마나 충족하는지를 단계적으로 평가하는 방법[4]이 제안되었고, LLM이 다른 LLM의 출력을 채점하는 자동 평가 체계[5]가 등장하면서 대규모 평가의 효율성이 높아졌다. 또한 코드 생성 과정에서 사실과 다른 코드를 생성하는 환각 현상을 유형별로 분류한 연구[6]도

*본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단 (RS-2025-00564249)의 지원을 받아 수행되었다.

† Corresponding Author

이루어졌다.

한편, LLM에게 부여하는 제약 조건의 수가 성능에 미치는 영향을 체계적으로 분석한 연구[3]와 모델 규모에 따라 능력이 구간별로 전환되는 현상을 정량화한 연구[8]가 보고되었다. 그러나 이들은 모델의 학습 조건이나 구조적 특성에 초점을 두고 있으며, 추론 시점에 프롬프트로 주입되는 보안 규칙의 양이 교정 성능에 어떤 구간별 영향을 미치는지는 아직 탐색되지 않았다.

2.2. 본 연구의 차별점

본 연구는 다음 두 가지 측면에서 기존 연구와 구분된다. 첫째, 모델 규모에 따른 능력 전환 현상[8]을 추론 시점의 규칙 제공량으로 확장하여, 보안 규칙의 양에 따라 교정 성능이 구간별로 달라지는 현상을 IaC 도메인에서 처음으로 정량화하였다. 둘째, 교정 에이전트가 최적 구간에서는 지시 조건 충족도[4]를 회복하고, 규칙 과다 구간에서는 정보 과부하로 인한 교정 실패[6]를 검토·재시도를 통해 줄이는 이중 역할을 동일 알고리즘 내에서 수행함을 입증하였다.

3. 제안방법

3.1. 교정 에이전트 (Negotiation Agent)

교정 에이전트의 핵심 원리는 의도 보존이다. 규칙을 위반한 코드를 전부 폐기하지 않고, 사용자의 원래 의도(예: 특정 자원 생성 요청)를 유지한 채 위반 부분만 선택적으로 수정하여 제안한다. 또한 이를 위해 에이전트는 네 단계로 동작한다. 먼저 원본 코드에서 사용자의 의도를 추출하고, 이전 시도에서 실패한 교정 방식을 기록하여 같은 실패를 반복하지 않도록 한다. 이어서 195개 보안 규칙에 기반한 규칙 검사와 Terraform 구문 검증을 순차적으로 수행한다. 이 과정을 최대 3회까지 반복하며, 검사를 통과할 때까지 교정안을 점진적으로 개선한다. 동일한 알고리즘이지만, 규칙 제공량에 따라 최적 구간에서는 교정 복구, 과다 구간에서는 오류 억제라는 서로 다른 행동이 나타난다.

3.2. 평가 지표

교정 성능은 Pass@1[7]을 기본 지표로 사용하였다. 각 시나리오에 대해 단일 시도로 교정에 성공했는지를 이진 판정하며, 이를 기반으로 지시 조건의 부분 충족률(SSR)과 완전 충족률(CSR)[4]을 함께 측정하였다. 교정 과정에서 모델이 원본에 없는 잘못된 코드를 생성하는 비율[6]도 별도로 추적하였다.

교정 결과의 실제 안전성은 산업 현장에서 사용되는 정적 분석 도구인 Trivy와 Checkov로 검증하였고, LLM이 교정 품질을 채점하는 자동 평가[5]도 보조 지표로 활용하였다. 평가자 간 일치도는 ICC(3,1)로 측정하였다.

통계적 검증에는 시나리오마다 난이도가 다른 점을 반영할 수 있는 혼합 효과 분석을 적용하고, 여러 구간을 한꺼번에 비교할 때 우연히 유의하게 나오는 경우를 걸러내는 보정을 수행하였다. 또한 효과 크기(Cohen's d)[9]를 함께 보고하였다. 이는 "통계적으로 유의하다"는 것과 별개로, 두 조건 사이의 성능 차이가 실제로 의미 있는 크기인지를 판단하기 위한 지표이다.

4. 실험 및 평가

4.1. 실험환경

본 실험은 AWS Bedrock(us-west-2) 환경에서 Claude Sonnet 4.6 모델을 사용하여 수행되었다. AWS Well-Architected Framework의 5대 원칙(보안, 비용, 성능, 안정성, 운영 우수성)에 기반한 24개 교정 시나리오를 구성하고, 규칙 제공 비율(RCR)을 0%에서 100%까지 11단계로 변화시키며 교정 전(Static)과 교정 후(Agent) 두 조건을 반복 측정하여 총 2,112건의 실험 데이터를 수집하였다.

외부 일반성 검증을 위해 실제 기업 환경의 취약 코드인 Bridgecrew TerraGoat 12개 시나리오(216건)와 학술 벤치마크인 IaC-Eval을 대상으로 추가 실험을 수행하였다. 또한 교정 에이전트의 구조적 이점을 검증하기 위해 단순 반복, 위반 피드백 재시도, 규칙 직접 삽입 등 대안 방식과의 비교를 포함한 648건의 통제 실험을 별도로 수행하였다. 세부 사항은 표 1과 같다.

항목	설정
컴퓨팅 (LLM 추론)	AWS Bedrock (us-west-2 cross-region)
LLM (main 생성)	Claude Sonnet 4.6 (Temperature=0.0)
LLM (cross-tier 검증, 408 runs)	Claude Sonnet 4.5 / Haiku 4.5 / Opus 4.7
LLM-as-Judge (동족, 144 paired)	Claude Haiku 4.5 + Sonnet 4.5
LLM-as-Judge (cross-family, 144)	Llama 3.1 70B Instruct (Meta) [5]
데이터셋 (main, 24)	AWS WAF 5-pillar Terraform IaC
데이터셋 (외부, 12 시나리오)	Bridgecrew TerraGoat (Apache 2.0)
산업 표준 스캐너	Trivy v0.70.0 (Aqua), Checkov v3.2 (Bridgecrew)
통계 분석	LME, FDR-BH, Cohen's d, Bootstrap CI (n=1,000)
Rule subset 규모	총 195 rules (Sec 80 / Cost 60 / Perf 55)
RCR 수준 (총 11)	0/20/30/40/50/60/70/80/100 (%)

표 1. 실험환경 LLM, 데이터셋, 산업 표준 스캐너, 통계 분석 및 사전 등록 구성을 요약한다.

4.2. 실험 결과 및 분석

4.2.1. Phase 구조 식별

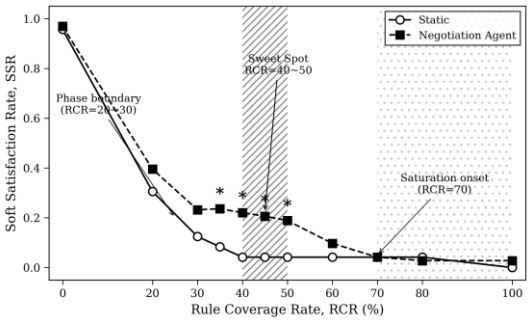

그림 1. RCR 증가에 따른 교정률[4] 변화를 보여준다.

Negotiation Agent는 Static 방식 대비 전반적으로 높은 SSR을 보였으며, 특히 RCR 40~50% 구간에서 가장 큰 성능 우위를 나타냈다. 반면 RCR 70% 이상에서는 성능이 포화되는 경향이 확인되었다.

4.2.2. Sweet Spot Recovery 효과

교정률은 RCR 증가에 따라 유의하게 감소하는 경향을 보였으며(LRT p=0.008), 특히 RCR 20~30% 구간에서 감소폭이 크게 나타났다. 반면 RCR 40% 이상에서는 성능 변화가 완만해지며, 포화 구간에 진입하는 것으로 확인되었다. 이에 따라 RCR 40~50% 구간을 Sweet Spot으로 선정하였다.

4.2.3. Saturation 구간에서의 Risk Reduction 효과

RCR=80% 포화 구간에서는 Pass@1 기준으로 교정 전후 간 유의한 차이가 나타나지 않았다(p=0.319). 그러나 잘못된 코드 생성 비율은 26.9%p 감소하였고(p<0.001), Trivy와 Checkov 기준 보안 오구성도 각각 98.7%, 84.6% 줄어들었다. 이는 동일한 교정 에이전트가 최적 구간에서는 성능 회복을, 포화 구간에서는 위험 감소를 제공하며, 규칙 제공량에 따라 효과의 양상이 달라짐을 보여준다.

4.2.4. 제안된 목표 지향적 보안 최적화 기법의 구성요소별 기여도 분석

Condition	*RCR=80	*RCR=40	**Δ
FULL	13.9%	37.5%	–
*w/o Intent	9.7%	12.5%	-25.0
*w/o failed_methods	8.3%	37.5%	0
w/o LLM-as-Judge	13.9%	50.0%	+12.5
max_attempts=1	5.6%	37.5%	0
LITE (n=32)	25.0%	50.0%	**+15.0**

표 2. Component ablation Pass@1, RCR=80

* pp: percentage point, RCR: Risk Compliance Rate.

** Δ: pp (Full 대비 성능 변화량), ** w/o: Without.

RCR=80(n=72), RCR=40(n=8), 및 LITE 직접 검증(n=32)을 대상으로 구성요소 제거 실험을 수행하였다. Intent 제거 시 RCR=40 조건의 Pass@1은 37.5%에서 12.5%로 감소하여 의도 보존이 Sweet Spot 구간의 핵심 요소임을 확인하였다. 반면 LLM-as-Judge를 제거한 구성과 LITE 구성은 FULL 대비 Pass@1이 12.5pp 향상되어, IaC 보안 환경에서는 결정론적 검증 체계만으로도 충분한 성능을 확보할 수 있음을 시사한다.

4.2.5. Cross-family LLM-as-Judge ICC Inflation

Anthropic 계열 모델 간 평가에서는 ICC(3,1) = 0.594가 관찰된 반면, Llama 3.1 70B를 포함한 교차 계열 평가에서는 0.317로 감소하였다. 이는 LLM-as-Judge에서 same-family bias 가능성을 시사한다. 그러나 본 연구는 Rule, Terraform, Trivy, Checkov 기반의 결정론적 지표를 주 평가 기준으로 사용하므로 해당 편향이 주요 결과에 미치는 영향은 제한적이다.

5. 향후 연구 및 결론

본 논문은 AWS 환경의 코드형 클라우드 인프라(IaC)를 대상으로 대규모 언어모델 기반 목표 지향적 보안 최적화 방법을 제안하고 그 효과를 실증적으로 분석하였다.

실험 결과, 규칙 제공 비율(RCR)에 따라 교정 성능이 단계적으로 변화하였으며, RCR 40~50% 구간에서는 교정률 향상 효과가, RCR 70% 이상 포화 구간에서는 Hallucination 및 보안 오구성 감소 효과가 확인되었다.

또한 동일한 의도 보존 기반 교정 메커니즘이 환경 조건에 따라 성능 회복과 위험 감소라는 서로 다른 효과를 보였으며, 결정론적 검증 체계가 존재하는 IaC 보안 환경에서는 LLM-as-Judge의 추가 기여가 제한적임을 확인하였다.

참고문헌

[1] Lewis, P., et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", NeurIPS, arXiv:2005.11401, 2020.

[2] Cui, J., et al., "OR-Bench: An Over-Refusal Benchmark for Large Language Models", ICML, 2024.

[3] Yao, S., et al., "COLLIE: Systematic Construction of Constrained Text Generation Tasks", ICLR, arXiv:2307.08689, 2024.

[4] Yan, Y., et al., "CodeIF: Benchmarking the Instruction-Following Capabilities of LLMs for Code Generation", ACL Industry Track, arXiv:2502.19166, 2025.

[5] Zheng, L., et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena", NeurIPS, 2023. arXiv:2306.05685.

[6] Liu, F., et al., "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation", arXiv:2404.00971, 2024.

[7] Chen, M., et al., "Evaluating Large Language Models Trained on Code", arXiv:2107.03374, 2021.

[8] Wei, J., et al., "Emergent Abilities of Large Language Models", TMLR, 2022.

[9] Cohen, J., "Statistical Power Analysis for the Behavioral Sciences", 2nd ed., Routledge, 1988.

*본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단 (RS-2025-00564249)의 지원을 받아 수행되었다.

† Corresponding Author