Enhancing Firewall Rule Anomaly Detection via LLM Alignment

Chang-Sheng Lee[1], I-Chen Lee[1,2], and Ling-Jyh Chen[1,3]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan
{johnsonlee911205, cclljj}@iis.sinica.edu.tw
ian52759@gmail.com
[2] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
[3] Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan

Abstract. Firewalls are a crucial part of modern network systems. However, managing a firewall with thousands of rules can be challenging. Rules anomalies can compromise security. Although algorithm-based methods are accurate and efficient at detecting anomalies, they cannot fully automate the correction of anomalies in datasets. Large Language Models (LLMs) provide a promising solution by enabling firewall tasks to be handled while maintaining user intent. To achieve anomaly correction, this paper first enhances the LLMs' understanding of anomalies by having them detect them. Our preliminary assessment shows that current state-of-the-art models have limited ability to generate and detect anomalies in firewall rules. This work explores several fine-tuning methods to address these limitations. We improved LLM performance on the firewall anomaly detection task through Supervised Fine-Tuning and Reinforcement Learning. Compared to methods without Fine-Tuning, the experimental results indicate that LLM alignment can improve model performance. These findings suggest that LLM alignment is an effective way to expand LLMs' knowledge. Future work could focus on improving the generalization of LLMs for detecting anomalies in larger firewall datasets.

Keywords: Firewall · Firewall Rule Anomaly · Large Language Model · Alignment · Supervised Fine-Tuning · Reinforcement Learning.

1 Introduction

With the rapid growth of the Internet, network security has become a critical concern. Firewalls are widely deployed to protect against cyberattacks by filtering unwanted packets. However, as the number of firewall rules increases, the configuration becomes increasingly complex and unintended errors may arise. These errors, referred to as "anomalies" by Al-Shaer and Hamed [1], occur when a single packet matches more than one rule. Such anomalies may invalidate the intended policies and compromise network security.

The severe consequences of anomalies underscore the importance of addressing this issue. Our goal is to generate concise and error-free firewall rule sets. Traditional tools such as Firewall Policy Advisor [1] and FIREMAN [2] have proven to be effective in detecting anomalies through algorithmic analysis. However, as Al-Shaer [3] observed, certain anomalies, such as generalization anomalies, cannot be eliminated without altering the intended behaviors. Consequently, anomaly correction cannot be fully automated by conventional methods.

Recent advancements in Large Language Models (LLMs) have sparked growing interest in applying artificial intelligence to network security. We explore the feasibility of using LLMs to address firewall rule anomalies. For an LLM to optimize rule sets, it must first understand the semantics of the firewall and reason about inconsistencies. Thus, error detection serves as a crucial intermediate step. In our previous work [4], we evaluated several state-of-the-art LLMs for anomaly detection. The results suggested that, while LLMs show potential, their precision and consistency are insufficient for complex firewall management. Addressing this gap requires alignment techniques to improve the reliability of the model. Wang et al. [5] reviewed approaches such as supervised fine-tuning (SFT) and reinforcement learning (RL), which are particularly effective in improving task performance. Building on this foundation, our study focuses on improving LLM performance in firewall anomaly detection through fine-tuning.

We selected a modern LLM and compared three training strategies: SFT, RL, and a hybrid of both. To support this, we developed a data generation program that can customize rules and anomalies. To ensure that performance differences are due to training rather than data distribution, we included the SFT dataset as a subset of the RL dataset. Finally, we evaluated the results of each method and parameter setting against a baseline.

The main contributions of this study are as follows.

- We designed diverse training and testing datasets based on a custom virtual firewall syntax reflecting real-world system fields.
- We demonstrated that fine-tuning significantly improves LLM performance in anomaly detection, as shown by comparisons of SFT, RL, and hybrid methods with baseline models.
- We identified optimal parameter settings for our dataset and documented key challenges encountered during training.

The remainder of this paper is organized as follows. Section 2 reviews firewall anomalies, detection methods, and LLM alignment techniques. Section 3 details the fine-tuning process. Section 4 presents the experimental results of the SFT, RL, and hybrid models. Section 5 concludes the paper and discusses future work.

2 Related Works

2.1 Firewall Rule Anomalies

A typical firewall functions as a packet filter, enforcing an access control list composed of rules defined by conditions and actions. Because packets are sequentially matched against rules until a match is found, the order of rules is

critical. If two rules overlap, their execution may depend on ordering, which can introduce unexpected behavior.

Al-Shaer and Hamed [1] classified anomalies that arise when two or more distinct rules match the same packet:

1. **Shadowing Anomaly:** A rule is shadowed when an earlier rule matches all packets that would otherwise match the shadowed rule, rendering it ineffective.
2. **Correlation Anomaly:** Two rules are correlated if each matches some packets that the other also matches, but not all.
3. **Generalization Anomaly:** A rule is a generalization of another if it matches all packets covered by a preceding specific rule.
4. **Redundancy Anomaly:** A redundant rule performs the same action on the same packets as another rule, such that removing it has no effect on the policy.

Importantly, anomalies are not always errors. They can be intentionally introduced. For example, a generalization anomaly often appears when a broad rule is modified by adding an exception.

2.2 Algorithm-Based Anomaly Detection

In addition to anomaly classification, Al-Shaer and Hamed also introduced the Firewall Policy Advisor [1], an algorithm for anomaly detection. The approach compares rule fields one by one, modeling each comparison as a state in a state diagram. After all fields are checked, the final state determines the anomaly type based on predefined logic.

Although effective, the Firewall Policy Advisor has limitations. It only detects anomalies pairwise and does so inefficiently. To improve efficiency, subsequent research proposed broader detection methods. Among these, FIREMAN [2] is a key contribution. Unlike pairwise approaches, FIREMAN detects anomalies in multiple rules by comparing each rule against all its predecessors. Whereas Firewall Policy Advisor requires $O(n^2)$ complexity, FIREMAN achieves $O(n)$ detection efficiency. However, this efficiency comes at the cost of not being able to identify which specific preceding rule caused the anomaly.

2.3 Evaluation of LLM-Based Anomaly Detection

To evaluate the feasibility of applying LLMs to firewall tasks, our previous research [4] assessed state-of-the-art models on anomaly-related tasks. We designed a two-stage experiment. In stage 1, four reasoning-focused LLMs generated firewall rule datasets containing anomalies. In stage 2, the same models attempted to detect anomalies in the generated datasets.

Analysis using confusion matrix metrics revealed that no model consistently performed well in both stages. This suggests that LLMs cannot yet be directly applied to real-world firewall management. Inaccurate anomaly detection could lead to overlooked risks and weaker security. Thus, additional training is essential if LLMs are to be applied to firewall anomaly detection.

2.4 LLM Alignment

LLMs are prone to hallucinations, such as generating false or irrelevant information. Alignment techniques, collectively called fine-tuning, are widely used to improve their reliability and ensure that the results meet human expectations. Wang et al. [5] summarized several prominent alignment strategies.

Supervised Fine-Tuning (SFT) trains a LLM to minimize cross-entropy loss against target answers, improving task adaptation and response quality. However, SFT primarily teaches the model to reproduce the best answers, limiting its ability to generalize beyond memorized patterns. Chu et al. [6] showed that SFT often relies on rote memorization and struggles with generalization across tasks.

Reinforcement Learning (RL) improves generalization by training a model through sequential actions and reward feedback. The LLM adjusts its parameters to maximize the cumulative rewards defined by a reward function aligned with the task objectives. Various RL algorithms have been proposed. Srivastava et al. [7] reviewed several, including Proximal Policy Optimization (PPO), a policy-gradient framework that iteratively updates model policies based on rewards from a learned reward model. Group Relative Policy Optimization (GRPO) extends this idea by generating multiple responses per prompt, normalizing their rewards within a group, and eliminating the cost of maintaining a separate value function estimator as required by PPO. In our experiments, we adopted GRPO to reduce memory usage and accelerate computation.

In practice, SFT and RL are often combined. A common example is Reinforcement Learning from Human Feedback (RLHF), which begins with SFT to stabilize outputs before applying RL for performance gains.

3 Methods

This study began by selecting an appropriate model, followed by constructing carefully designed datasets for training and evaluation. We then fine-tuned the model using three approaches: supervised fine-tuning (SFT), reinforcement learning (RL), and a hybrid of the two, and finally evaluated the trained models on a testing dataset.

3.1 Model Selection

We considered the following criteria when selecting a model:

1. **Open-source:** While commercial models such as OpenAI o3 generally outperform open source models, most commercial options do not allow fine-tuning.
2. **Reasoning-oriented:** Models that extend reasoning time and decompose problems into smaller steps tend to achieve better performance in complex tasks.

3. **Sufficient parameter size:** Li et al. [8] demonstrated that models with fewer than 3B parameters cannot fully benefit from long chain-of-thought reasoning or knowledge distillation. Similarly, TinyZero [9], which applies PPO to math tasks, showed that models with 1.5B parameters lack effective self-reflection capabilities.

Based on these considerations, we selected Qwen3-4B, a leading reasoning-focused model as of September 2025. We fine-tuned the base version to ensure that training outcomes were not confounded by pre-applied adaptations.

3.2 Datasets

Due to the lack of standardized firewall datasets and the diversity of firewall syntaxes (e.g., Linux iptables [10] and Fortinet Fortigate [11]), we designed a JSON schema inspired by Fortinet Fortigate fields. This abstraction is shown in Fig. 1.

```
{
  {
    "counter": <rule index>,
    "srcintf": ["<source interfaces>"],
    "dstintf": ["<destination interfaces>"],
    "srcaddr": ["<source address (IP/CIDR)>"],
    "dstaddr": ["<destination address (IP/CIDR)>"],
    "action": "<accept or deny>",
    "service": ["<protocol:port(s)>"]
  },
  ...
}
```

Fig. 1. Firewall rule fields represented in hierarchical JSON.

After defining the rule format, we implemented a dataset generator that produces scalable datasets with configurable numbers of rules, anomalies, and field values. However, for this experiment, we restricted the datasets to one pair of rules, as larger sets introduce additional complexity that obscures the effects of variable levels on model performance. To improve generalization, we introduced controlled variations in the datasets. All types of anomaly (including "none") were balanced. The field values were randomized as follows:

- `srcintf` and `dstintf`: randomly selected from the common interface options.
- `srcaddr` and `dstaddr`: CIDR or IP masks ranging from /8 to /32, with one to three values chosen randomly.

– `service`: TCP or UDP, with ports specified as three individual ports or as ranges.

We also developed a brute-force anomaly detection program to generate ground-truth labels. Both generation and detection programs are available upon request.

Using this framework, we created three datasets: one for SFT, one for RL, and one for testing. The prototype datasets consisted of two fields: *rule_ pair* and *answer*. For SFT, we added an additional *thought* field, generated via xAI Grok3 API using the prompt shown in Fig. 2. This dataset contained 1000 entries due to resource constraints. The RL dataset, which required more extensive exploration, contained 10,000 entries, including the 1000 from SFT to allow fair comparison. The test dataset contained 1000 entries and did not overlap with either training set. All datasets are available at `https://huggingface.co/johnson1205`.

Prompt: You are an expert system tasked with analyzing firewall rulesets to detect anomalies. Please tell me why the following rules exhibit anomaly anomaly.

Fig. 2. Prompt used to generate chain-of-thought reasoning in the SFT dataset.

3.3 Prompt and Output Template

Following DeepSeek-R1 [12], we constrain the model output to a structured format that separates reasoning from final predictions. The intermediate reasoning was enclosed between `<THINK>` and `</THINK>`, while final answers were enclosed within `<SOLUTION>` and `</SOLUTION>`.

The complete prompt, shown in Fig. 3, combined chain-of-thought prompt [13] with a few-shot examples defining each type of anomaly. The same prompt was applied during training and inference to reinforce consistent reasoning and structured outputs.

3.4 Supervised Fine-Tuning

In the SFT dataset, *thought* entries were enclosed with `<THINK>` and `</THINK>`, while *answer* entries were enclosed with `<SOLUTION>` and `</SOLUTION>`.

The parameter configurations are shown in Table 1. The learning rate was systematically varied, as it is critical to balance convergence stability and training efficiency. Controlled experiments were conducted at rates from 1×10^{-4} to 5×10^{-4} to identify the optimal setting. Training was performed on the SFT dataset, followed by evaluation on the testing dataset. All training and evaluation codes are available at `https://github.com/johnson1205/LLM_Firewall_Anomaly`.

system prompt: You are given a problem. Think about the problem and provide your working out. Place it between `<THINK>` and `</THINK>`. Then, provide your solution between `<SOLUTION></SOLUTION>`.

user prompt: You are an expert system tasked with analyzing firewall rulesets to detect anomalies. The firewall rules are provided as a dataset of JSON objects, each with the following fields:
- 'srcintf': Source Interface
- 'dstintf': Destination Interface
- 'srcaddr': Source Address (IP/CIDR)
- 'dstaddr': Destination Address (IP/CIDR)
- 'service': Protocol and Service (format: 'protocol:service')
- 'action': Action ('allow' or 'deny')

You must detect the following anomalies step-by-step for each rule pair in the provided dataset:
1. **Shadowing Anomaly**
Definition: Rule R2 is shadowed by a previous Rule R1 if:
- Action of R1 $\neq$ Action of R2
- R1 fully matches all packets that R2 would match (R1 fields encompass or equal corresponding fields in R2)

If a shadowing anomaly is detected, output: `<SOLUTION>shadowing</SOLUTION>`
2. **Generalization Anomaly**
Definition: Rule R2 is a generalization of Rule R1 if:
- Action of R1 $\neq$ Action of R2
- R2 matches all packets matched by R1 (R2 fields are broader or equal to corresponding fields in R1)

If a generalization anomaly is detected, output:
`<SOLUTION>generalization</SOLUTION>`
3. **Correlation Anomaly**
Definition: Rules R1 and R2 are correlated if:
- Action of R1 $\neq$ Action of R2
- R1 matches some packets R2 matches, and R2 matches some packets R1 matches (partial overlap)

If a correlation anomaly is detected, output:
`<SOLUTION>correlation</SOLUTION>`
4. **Redundant Anomaly**
Definition: Rule R2 is redundant if:
- Action of R1 = Action of R2
- R1 fully matches all packets matched by R2, rendering R2 unnecessary.

If a redundant anomaly is detected, output: `<SOLUTION>redundant</SOLUTION>`
5. **None (No Anomaly)**
Definition: If none of the above anomalies are detected between the rule pair. If no anomaly is detected, output: `<SOLUTION>none</SOLUTION>`
{Rule_pair}

Fig. 3. Prompt template used for SFT training.

Table 1. Parameters for SFT training.

Parameter	SFT
Gradient Accumulation Steps	1
Warmup Steps	50
Epoch	2
Weight Decay	0.01
Optimizer	Adam
Seed	3407

3.5 Reinforcement Learning

For RL, we adopted the GRPO algorithm, which generates four candidate responses per input and averages their rewards. This eliminates the need for a separate value function, reduces computational overhead, accelerates convergence, and improves accuracy.

The parameter and sampling settings are shown in Tables 2 and 3.

Table 2. Parameters for RL training.

Parameter	RL
Learning Rate	5e-6
Gradient Accumulation Steps	1
Warmup Steps	100
Epoch	1
Weight Decay	0.01
Optimizer	Adam
Temperature	1.0

Table 3. Sampling parameters.

Parameter	Value
Min_p	0.1
Top_p	1.0
Top_k	-1
Seed	3407

To encourage structured and accurate output, we define four reward functions:

- **match_format_exactly:** Verifies strict compliance with the required structure. Rewards 1.0 if they are exactly matched.
- **match_format_approximately:** Reward partial compliance with the required structure to encourage incremental progress. Reward 1.0 for each tag that appears exactly once.
- **answer_length_score:** Penalizes overly verbose outputs. Full reward (score 1.0) for ≤ 15 characters, linearly decreasing reward (score $(100 - \text{length})/85$) for 16 to 100 characters, and zero for > 100.
- **check_answer:** Verifies semantic correctness and strict formatting. Scored on a graded scale (1.0 for exact, 0.7 for correct but misformatted, down to 0.0 for invalid outputs).

This multilevel scoring ensured that models were rewarded for both semantic accuracy and structural discipline, enhancing reliability in deployment.

3.6 Hybrid Fine-Tuning

SFT provides structured learning of anomaly definitions and output formats, while RL enhances adaptability through reward-driven optimization. To take advantage of both, we applied a two-stage hybrid strategy:

1. SFT was first applied to stabilize the model reasoning structure and output.
2. The SFT-adapted model was then further optimized with RL to refine the predictions using custom rewards.

Because the RL dataset intentionally included the SFT dataset, the performance differences reflect training paradigms rather than data distribution. To further validate this, we evaluated the hybrid model in both the testing dataset and the SFT dataset.

4 Experiments and Results

4.1 Setup

The experiments were carried out on RunPod [14] using a single NVIDIA RTX 6000 Ada GPU (48 GB VRAM). To mitigate hardware constraints, we adopted the Unsloth fine-tuning framework [15], which accelerates training and reduces VRAM usage through kernel-level optimizations while reporting no accuracy regression.

4.2 Baseline

To establish a baseline, we evaluated the pretrained model on the testing dataset without any task-specific tuning. The model achieved a precision of 0.359, providing a reference point for subsequent improvements and indicating that Qwen3 has limited prior knowledge of firewall anomaly detection.

4.3 Supervised Fine-Tuning

We trained and evaluated SFT models on learning rates from 1e−4 to 5e−4. The results are summarized in Table 4.

Table 4. SFT accuracy on the testing dataset across learning rates.

Learning Rate	SFT
1e-4	0.544
2e-4	0.922
3e-4	0.978
4e-4	0.982
5e-4	0.986

All SFT settings substantially outperform the untuned baseline, confirming that fine-tuning is effective for this task. Performance improves monotonically up to 5e−4. For completeness, we also trained at 6e−4 and observed a lower accuracy of 0.960, suggesting the onset of instability from overly aggressive updates. Therefore, we consider 5e−4 to be the optimal learning rate in our dataset.

4.4 Reinforcement Learning

After 10,000 RL steps with GRPO, the model reached an accuracy of 0.992 in the testing dataset. As with SFT, RL produces significant gains over the baseline. The slight edge over SFT is plausibly attributable to the larger RL dataset and the exploration dynamics.

We visualized training with Weights & Biases [16]. Figure 4 shows the total reward versus the training steps. During training (first $\sim$ 500 steps), rewards are unstable as the model learns the output format. Once tag usage stabilizes, reward improvements are predominantly driven by semantic correctness, with noticeable convergence inflections around the 2,000 and 5,000 steps.

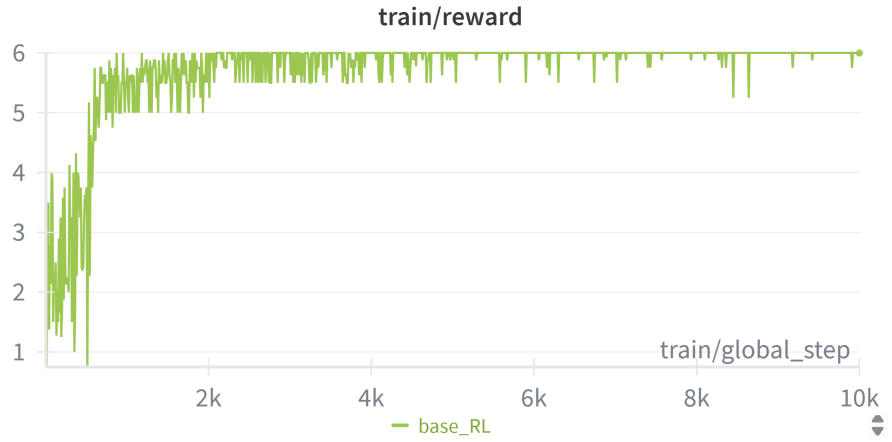

Fig. 4. RL total reward as a function of training steps.

Figure 5 depicts the completion length (response length) over steps. After $\sim$500 steps, explicit reasoning rapidly diminishes. One explanation is reward hacking: because we did not positively reward a minimum reasoning length, the policy found short, valid outputs that maximize reward. Another possibility is a shift from explicit to implicit reasoning while maintaining accuracy. Further analysis of this behavior is left to future work. Training logs are available at `https://wandb.ai/johnsonlee911205-academia-sinica/huggingface?nw=nwuserjohnsonlee911205`.

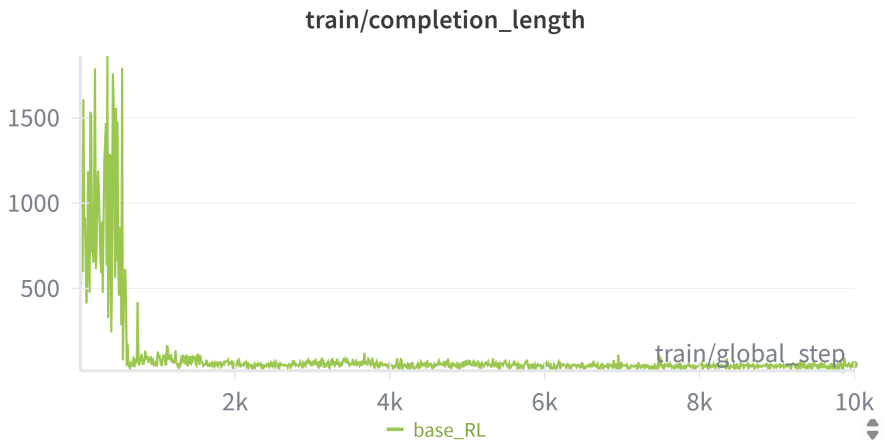

Fig. 5. RL completion length over training steps.

4.5 Hybrid Fine-Tuning

Oriented by the SFT study, we continued training on RL with the SFT checkpoint trained at the learning rate 5e−4. Upon evaluation of both the testing dataset and the SFT dataset, the hybrid model achieved accuracies of 0.994 and 0.993, respectively. The near-identical performance on unseen (testing) and seen (SFT) distributions indicates a strong generalization.

Figure 6 plots hybrid training rewards. The curve closely resembles pure RL, probably because the SFT-initialized model already adheres to the output format, leaving semantic refinement as the primary driver of reward increases. The hybrid of SFT and RL shows noticeable convergence inflections around the 2,000 and 4,000 steps. Compared to pure RL, the hybrid approach achieves a more effective convergence of content rewards.

5 Conclusion

We investigated three alignment strategies, SFT, RL, and a two-stage hybrid, using custom datasets for firewall anomaly detection. All methods substantially improved accuracy over the untuned baseline (from 0.359 to as high as 0.994), with RL and the hybrid approach performing best. We also analyzed the sensitivity to the learning rate in SFT and identified 5e−4 as optimal for our setting, while higher rates (e.g., 6e−4) degraded the precision.

A notable observation is the collapse of explicit reasoning length during RL, which may reduce interpretability even as accuracy remains high. Future work will stabilize training while preserving applicable intermediate reasoning, scale to larger rule sets, and explore curriculum learning to introduce multi-rule scenarios without catastrophic forgetting progressively.

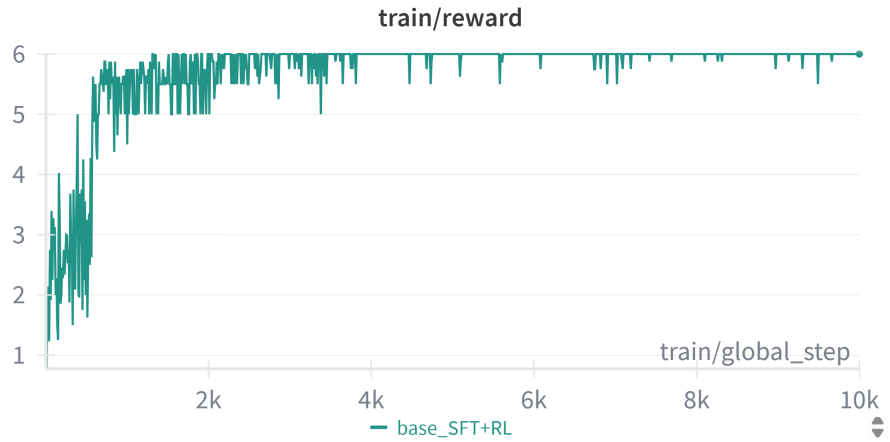

Fig. 6. Hybrid (SFT→RL) total reward over steps.

References

1. Al-Shaer, E., Hamed, H.: Firewall policy advisor for anomaly discovery and rule editing. In: IFIP/IEEE Eighth International Symposium on Integrated Network Management, 2003. pp. 17–30 (2003). https://doi.org/10.1109/INM.2003.1194157
2. Yuan, L., Chen, H., Mai, J., Chuah, C.N., Su, Z., Mohapatra, P.: Fireman: a toolkit for firewall modeling and analysis. In: 2006 IEEE Symposium on Security and Privacy (SP'06). pp. 15 pp.–213 (2006). https://doi.org/10.1109/SP.2006.16
3. Al-Shaer, E.: Automated Firewall Analytics: Design, Configuration and Optimization (01 2014). https://doi.org/10.1007/978-3-319-10371-6
4. Lee, C.S., Chen, L.J.: Can ai outsmart firewall errors? a study on llms for anomaly generation and detection. In: 2025 IEEE Conference on Dependable and Secure Computing (DSC) (2025), accepted (to appear)
5. Wang, Y., Zhong, W., Li, L., Mi, F., Zeng, X., Huang, W., Shang, L., Jiang, X., Liu, Q.: Aligning large language models with human: A survey (2023), `https://arxiv.org/abs/2307.12966`
6. Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q.V., Levine, S., Ma, Y.: Sft memorizes, rl generalizes: A comparative study of foundation model post-training (2025), `https://arxiv.org/abs/2501.17161`
7. Srivastava, S.S., Aggarwal, V.: A technical survey of reinforcement learning techniques for large language models (2025), `https://arxiv.org/abs/2507.04136`
8. Li, Y., Yue, X., Xu, Z., Jiang, F., Niu, L., Lin, B.Y., Ramasubramanian, B., Poovendran, R.: Small models struggle to learn from strong reasoners (2025), `https://arxiv.org/abs/2502.12143`
9. Pan, J., Zhang, J., Wang, X., Yuan, L., Peng, H., Suhr, A.: Tinyzero. https://github.com/Jiayi-Pan/TinyZero (2025), accessed: 2025-01-24
10. netfilter. firewalling, nat, and packet managing for linux. `https://www.netfilter.org/`, accessed: 2025-09-07

11. Fortinet Inc.: Fortigate / fortios 7.6.1 administration guide: Firewall policy. https://docs.fortinet.com/document/fortigate/7.6.1/administration-guide/656084, accessed: 2025-09-07
12. DeepSeek-AI: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning (2025), https://arxiv.org/abs/2501.12948
13. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022)
14. RunPod Inc.: Runpod documentation. https://docs.runpod.io/overview, accessed: 2025-09-07
15. Daniel Han, M.H., team, U.: Unsloth (2023), http://github.com/unslothai/unsloth
16. Biewald, L.: Experiment tracking with weights and biases (2020), https://www.wandb.com/, software available from wandb.com