January 2024

# Automatically Detecting Expensive Prompts and Configuring Firewall Rules to Mitigate Denial of Service Attacks on Large Language Models

Assaf Namer

Prashant Kulkarni

Erik Jeansson

Brandon Maltzman

Hauke Vagts

# Automatically Detecting Expensive Prompts and Configuring Firewall Rules to Mitigate Denial of Service Attacks on Large Language Models

ABSTRACT

Denial of service attacks on generative artificial intelligence systems, e.g., large language models (LLMs), can include sending LLMs requests that include expensive prompts designed to consume computing resources and degrade model performance. This disclosure describes techniques to automatically detect such prompts and then configure firewall rules that prevent such prompts in subsequent requests from reaching the LLM. Per the techniques, prompts provided to an LLM are matched against input and output token size as well as resource utilization to identify prompts that deviate significantly from a baseline. Expensive prompts are identified, and semantically similar prompts are automatically generated using the same LLM or another model. A subset of the generated prompts that are semantics similar to expensive prompts are identified by comparing respective vector embeddings. The subset of prompts and the received expensive prompts are provided to a pre-trained LLM that generates firewall rules, e.g., web application firewall (WAF) rules. Incoming requests from applications are evaluated based on the rules, and expensive prompts are blocked from reaching the LLM or are rate-limited.

KEYWORDS

- Expensive prompts
- Model Denial of Service (MDoS)
- Large language model (LLM)
- Web application firewall (WAF)
- MLOps
- SecOps
- LLM security
- Generative AI (Gen AI)
- Semantically similar prompt
- Resource utilization

BACKGROUND

A threat faced by providers of large language models (LLMs) is model denial of service (MDoS) attacks. MDoS is a type of cyber-attack where machine learning models are marked unavailable by overloading the model provider with a high volume of crafted inputs designed to waste resources and degrade model performance. These attacks are characterized by repeatedly sending long, meaningless text blocks to exhaust processing quotas (e.g., GPU); posing confusing questions or paradoxes to reduce output quality over time; entering prompts (known as expensive prompts) that are crafted to cause the model to produce a large amount of text in the response; etc. Model denial of service attacks are similar to domain name system (DNS) amplification attacks. The risks of sustained denial of service include increased costs, service level agreement violations, rapid surge pricing, inflated resource consumption, data egress fees, budget overruns, etc. While the total number of output tokens per response may be subjected to a cap (e.g., up to 1024 tokens), as models become integrated in more applications, the need for outputs with greater token size increases, thus limiting the capping-based security approach.

Models can be protected against MDoS and other input-based attacks by proactively identifying and blocking anomalous patterns. Some of the techniques used to mitigate risks of MDoS include:

- **Rate limiting:** Throttling requests to prevent overload and abuse of model servers.
- **Request throttling:** Slowing down or blocking unusual spikes in traffic to stabilize workloads.
- **Input filtering:** Scanning inputs for known indicators of exploitation like excessively long text.

- **Load balancing:** Distributing queries across servers to smoothly handle increased loads.
- **Auto-scaling:** Automatically provisioning additional model servers based on demand to maintain performance.
- **Service quotas:** Setting usage quotas per customer account to contain costs of attacks.
- **AI-assisted queue prioritization:** Using AI to score request urgency and priority to triage service.
- **Model degradation:** Gradually reducing model capabilities or precision to spread load if needed.
- **Content Delivery Network (CDN) caching:** Caching common model responses at edge locations to absorb spikes in traffic.
- **Client-side mitigations:** Software development kit (SDK) level features such as retry logic, graceful degradation, and input validation.

These techniques have several limitations in addressing the risk of MDoS attacks. Some of these defenses are reactive and vulnerable to circumvention. Some of the techniques are relatively expensive and slow to implement in production. Queue prioritization (rather than first-come ordering) based on predictive confidence scores can help to ensure critical queries get through. However, completely locking down models for other queries has a substantial impact on the intended users and business outcome.

## DESCRIPTION

This disclosure describes techniques to detect abuse of large language models (LLMs) by measuring input and output size over time, detect anomalies in GPU/TPU utilization, and apply policies on the downstream channel(s). The techniques include identifying expensive

prompts and generating semantically similar prompts to be used for web application firewall (WAF) rules. An embedding database is used to extend expensive prompt rule creation by identifying expensive prompts that are semantically similar. WAF rules can be generated for automatic actions to be taken, for example, blocking, redirecting, rate limiting, etc., on identified expensive input prompts, where the rule applies to a category of expensive prompts rather than just the original one. WAF rules are configured on the basis of original and semantically similar prompts ensuring a wide scope of identification. Input prompts received during model serving are scored based on similarity to pre-generated WAF rules. If a received prompt is above a threshold, specific actions are triggered. Rule configuration can be varied as appropriate. The actions are triggered as soon as it is determined that the input prompt matches a WAF rule, e.g., if the first N tokens of the prompt match the WAF rule, rather than matching the entire prompt against WAF rules.

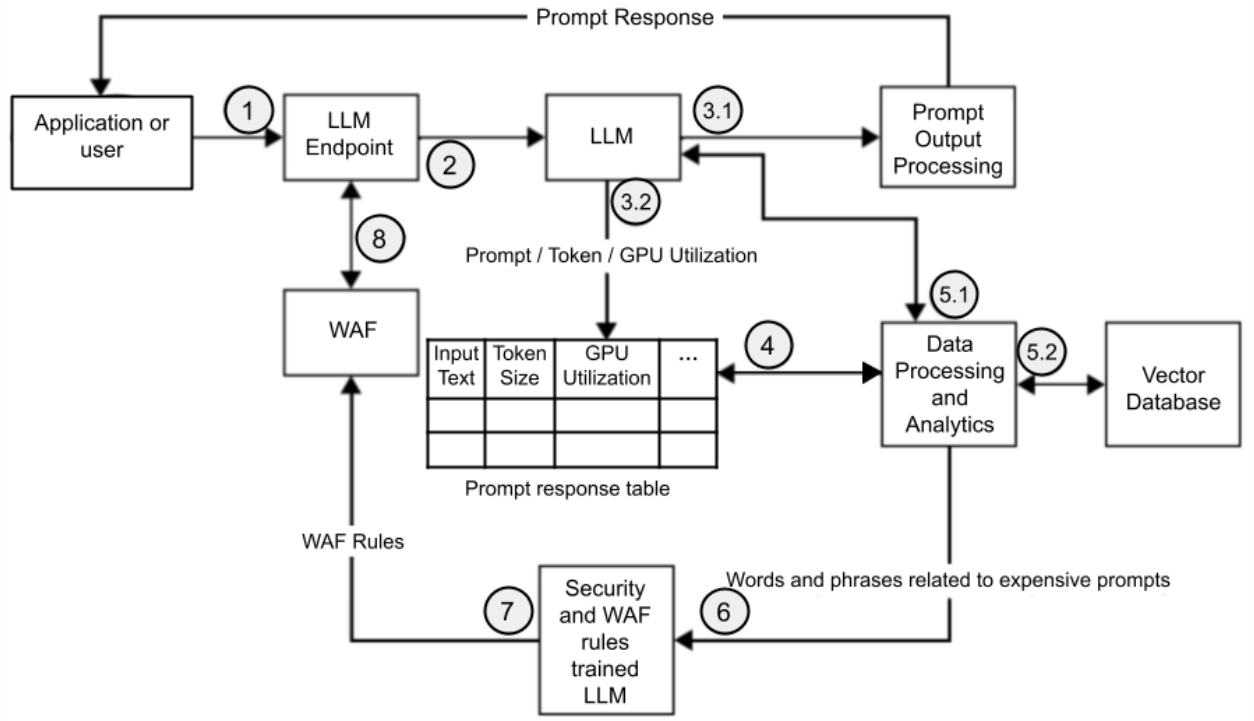


**Fig. 1: Automatically generating web application firewall (WAF) rules**

Fig. 1 illustrates the process of identifying expensive prompts, generating semantically similar prompts, and utilizing a model to generate WAF rules that can be applied to proactively identify expensive prompts received at the LLM endpoint that receives prompts from applications or users.

As illustrated in Fig. 1, a large language model (LLM) endpoint receives (1) an application programming interface (API) request from an application or user. The LLM endpoint can be a load balancer or any other system that can process and send API requests to backend services. The LLM endpoint is connected to a web application firewall (WAF) that can evaluate Layer-7 HTTP/S requests. The WAF is used to enforce security policies based on predefined rules, which can be of different types and can use a rules framework.

The LLM endpoint uses (8) the WAF to evaluate the API request. If the request satisfies the rules (e.g., does not include an expensive prompt or other types of prompts that violate a rule), the request is sent (2) to a large language model (LLM). Based on the input prompt, the LLM generates a response that is subjected to (3.1) prompt output processing. After the output processing (e.g., filtering, evaluation, etc.), the response is sent back to the application.

The response is also sent (3.2) to a table that stores the prompt and the number tokens or characters in the LLM response. In the context of LLMs, tokens are basic units of text (e.g., words, subwords, characters, or larger linguistic units), code, or other types of data the LLM uses to process and generate the response. Additionally, resource utilization details for resources such as graphics processing units (GPUs) or machine learning processors for generating the response to the prompt are obtained using resource monitoring tools and are also stored (3.2) in the table. Such data may be stored only for prompts that are deemed expensive.

Metadata from each prompt is stored in the table. The table can be sorted by prompt expensiveness (i.e., the number of tokens and/or resource utilization).

A set of processes and algorithms are utilized to analyze (4) data stored in a table as part of the data processing and analytics stage. Various parameters, such as average, mean, top percentile, etc., are calculated from the data in the table. The expensiveness of the prompt can be determined based on a comparison of the values in the table against the baseline of resource utilization and token mean. For example, a token may be determined to be an expensive token if it has a resource utilization which is two times (or other suitable value) the standard deviation from the mean. If the resource utilization for a prompt has a deviation more than the threshold resource utilization for the number of tokens, it can be identified as an expensive prompt. The baseline can also be set based on risk tolerance and application usage.

Once a prompt is identified as expensive, a request is sent to the LLM to generate (5.1) semantically similar prompts. This can be implemented using the same LLM used for application (as depicted in Fig. 1) or a dedicated, smaller LLM. A generated set of similar prompts is received from the LLM (5.1). The newly generated prompts are tested (5.2) against a vector database to verify a similar embedding space distance between each newly-generated prompt and the original expensive prompt (5.2). The most semantically similar prompts are then selected.

For example, consider a received input prompt: “Write a fantasy novel about a young wizard embarking on an epic quest”. This is a relatively expensive prompt, both in output tokens and resource utilization, since it requires generating a large amount of text (as specified by the presence of the term “novel” in the prompt) in the response. The generated similar prompts may then be as follows:

- “Compose a fantasy story regarding a young sorcerer beginning an adventurous expedition”
- “Create a tale of magic focused on a novice mage setting out on a mythic journey”
- “Draft a magical fiction about a fledgling warlock undertaking a formidable mission”
- “Produce a fantasy book concerning a burgeoning enchanter venturing on an ambitious crusade”
- “Assemble a fairy tale volume regarding an inexperienced magician departing on a formidable endeavor”

These prompts are turned into vector embeddings to enable the measurement of similarity distance and identifying similar prompts. The original expensive prompt and the selected similar prompts are sent (6) to a security LLM model (or other generative model) that is pre-trained to build and optimize web application firewall (WAF) rules. Alternatively, these rules can also be created manually, e.g., by a security team. WAF rules can be specific to the entire prompt or can be broader, e.g., using keywords from the provided prompts. The generated WAF rules are deployed (7) on the WAF, thereby enforcing a tailored security policy that includes dynamically updated rules generated based on identified expensive prompts and semantically similar prompts. When a new expensive prompt is received, WAF rules can, thus, automatically block or rate limit the request from the application or take other suitable action based on enterprise settings.

In this manner, expensive prompts that are semantically similar to prior prompts are blocked or rate-limited from reaching the LLM, thus saving processing resources. WAFs are optimized to evaluate incoming requests from applications and can be more efficient than other mechanisms such as data loss prevention (DLP) or pre-trained LLMs that evaluate received

requests. Some advantages of applying security control downstream from the LLM include the following:

- Flexibility and scalability, which allows for easier adaptation to changes in technology, user requirements, or emerging threats;
- Minimal system impact;
- Rapid deployment of security updates; and
- Reduced development costs and time.

The described techniques can be applied to secure LLMs or other types of generative models against attacks, including MDoS attacks. The techniques can protect LLMs from malicious requests from applications and users that use LLMs for natural language processing tasks. The techniques ensure availability and deployability of the model with pay-per-use pricing for API requests or other forms of usage-based billing. MDoS attacks that can result in a spike in resource usage and high cost are detected and mitigated downstream from the model.

The described techniques can be implemented relatively easily as a managed service for cloud computing customers. The techniques enhance model security and provide customers with a flexible, scalable way to minimize impact of model attacks, including controlling costs. The techniques enable cloud customers that use managed cloud services to build their own model protection.

## CONCLUSION

Denial of service attacks on generative artificial intelligence systems, e.g., large language models (LLMs), can include sending LLMs requests that include expensive prompts designed to consume computing resources and degrade model performance. This disclosure describes techniques to automatically detect such prompts and then configure firewall rules that

prevent such prompts in subsequent requests from reaching the LLM. Per the techniques, prompts provided to an LLM are matched against input and output token size as well as resource utilization to identify prompts that deviate significantly from a baseline. Expensive prompts are identified, and semantically similar prompts are automatically generated using the same LLM or another model. A subset of the generated prompts that are semantics similar to expensive prompts are identified by comparing respective vector embeddings. The subset of prompts and the received expensive prompts are provided to a pre-trained LLM that generates firewall rules, e.g., web application firewall (WAF) rules. Incoming requests from applications are evaluated based on the rules, and expensive prompts are blocked from reaching the LLM or are rate-limited.

REFERENCES

1. "HackerOne and the OWASP Top 10 for LLM: A Powerful Alliance for Secure AI" available online at https://www.hackerone.com/vulnerability-management/owasp-llm-vulnerabilities#:~:text=The%20supply%20chain%20in%20LLMs,and%20even%20complete%20system%20failures accessed Jan 18, 2024.
2. "5 Approaches to Solve LLM Token Limits," available online at https://deepchecks.com/5-approaches-to-solve-llm-token-limits/ accessed Jan 18, 2024.
3. "What is a Vector Database & How Does it Work? Use Cases + Examples," available online at https://www.pinecone.io/learn/vector-database/ accessed Dec 13, 2023
4. "OWASP Top 10 for LLM 2023: Understanding the Risks of Large Language Models," available online at https://www.giskard.ai/knowledge/owasp-top-10-for-llm-2023-understanding-the-risks-of-large-language-models accessed Dec 13, 2023

5. "DNS amplification DDoS attack | Cloudflare" available online at https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/ accessed Jan 18, 2024.
6. "reCAPTCHA firewall policies overview | reCAPTCHA Enterprise | Google Cloud" available online at https://cloud.google.com/recaptcha-enterprise/docs/firewall-policies-overview#policy-actions accessed Jan 18, 2024.
7. "Google Cloud Armor," available online at https://cloud.google.com/security/products/armor?hl=en accessed Jan 18, 2024.
8. "OWASP ModSecurity Core Rule Set," available online at https://owasp.org/www-project-modsecurity-core-rule-set/ accessed Jan 18, 2024.
9. "System Management Interface SMI," available online at https://developer.nvidia.com/nvidia-system-management-interface accessed Jan 18, 2024.
10. "Sklearn.metrics.pairwise.cosine_similarity," available online at https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html accessed Jan 18, 2024.
11. "Supercharging security with generative AI," available online at https://cloud.google.com/blog/products/identity-security/rsa-google-cloud-security-ai-workbench-generative-ai accessed Jan 18, 2024.
12. "Foundation models," available online at https://cloud.google.com/vertex-ai/docs/generative-ai/learn/models#foundation_models accessed Jan 18, 2024.
13. "New AI capabilities that can help address your security challenges | Google Cloud Blog" available online at https://cloud.google.com/blog/products/identity-security/security-ai-next23 accessed Jan 18, 2024.