
Durinn Calibration: Using OWASP Aware Distributions to Enhance Code Vulnerability Detection

Ryan Marinelli
University of Oslo

Victor Strandmoe
Durinn AS

Abstract

Hacktoberfest is a yearly tradition organised by DigitalOcean to promote contributions to open source code. It's seen among software development folks as an opportunity to give back to the community. The Durinn team investigated the drift of OWASP Top Ten issues for 60 repositories during this period to see if it was possible to classify code snippets as “secure” based on their contribution to the resolution of vulnerabilities. The analysis led to the creation of two datasets. One tokenised vulnerable code snippets, the other high-level indicators associated with code repositories. The team used two other well-known datasets to evaluate a calibration of a prompt-injection inference model by ProtectAI, `deberta-v3-base-prompt-injection-v2`. This model is promoted by workflow automation platforms as providing guardrails for LLM inputs.

In 2025, guardrails have been adopted in all services that utilise LLM. Their implementation, however, varies on company maturity and the requirements of the domain. Assistants, irrespective of domain, train their models on existing open source datasets too making MLOps susceptible to model poisoning (OWASP Foundation). Assistants' focus on efficacy and LLM guardrails being limited to looking at the input and output of the model made us curious to how attune well-known prompt-injection models would be to identify vulnerabilities from a well-known poisoned

dataset. In a world of increasing agent interoperability, this is an important function for ensuring agent-uptime (Boisvert).

The calibrated model showed a significant improvement in the number of vulnerabilities identified but an increase in the number of false-positives that is expected with model calibration.

In a world of increasing agent interoperability, maintaining reliable and continuous agent performance has become essential. As autonomous systems exchange information and coordinate decisions, their ability to surface accurate security signals directly influences overall agent-uptime and operational safety (Boisvert). Calibration plays an important role here: rather than modifying the underlying model, it adjusts how the model’s outputs are interpreted so that agents can make more stable and risk-aware decisions (Guo et al.).

The calibrated model showed a substantial improvement in the number of vulnerabilities correctly identified, confirming calibration as an effective way to enhance sensitivity in a lightweight manner. This improvement was accompanied by an increase in false positives—an outcome that is expected in most calibration settings, where shifting the decision threshold makes the model more responsive to weaker signals (Niculescu-Mizil and Caruana). Even so, the overall indication demonstrates that calibration potentially could sharpen a model’s ability to detect potential security issues in an agent-driven environment.

Keywords: AI security, AI code, code assistant, poisoning

1. Introduction

The goal of this research endeavour is to provide a macro-analysis for the open source community in celebration of a month-long period of meaningful contributions to coding libraries gratefully utilised by millions of people.

The development of LLMs and open-source have become intertwined in that people develop features and datasets and are used to reinforce the ecosystem by having LLMs and agents learn and enrich themselves on the content in a positive feedback loop. One of the major events that marks the open source calendar is Hacktoberfest. It is a celebration of the community to develop and support projects throughout the month of October. These contributions support not only the projects but the learning of models and the agentic tools that benefit inadvertently. However, not all the contributions are of the best quality. Hacktoberfest is a learning opportunity with a set deadline. People are prioritizing making their mark rather than making it maintainable. Thus, in order to ensure models learn the richest examples, this work

proposes a new dataset constructed grounded in scans of the contributions of Hacktoberfest. This dataset is then used to inform a classifier to filter out potentially problematic sources of training data to ensure the latest models learn from the richest data and avoid problematic pollution.

1.1 Contributions

The first contribution of this work is the creation of Durrinn’s Hacktoberfest Retrospective dataset. This dataset is designed to support the fine tuning and calibration of filtering agents to safeguard data diets.

The second contribution is the application of the dataset to calibrate state of the art models. Deberta is a model constructed by Microsoft and is specialized for classification tasks like filtering. This has been finetuned by ProtectAI to guard against problematic inputs to models with a focus on prompt injection. By calibrating their model with the Hacktoberfest Respective, the model is more accurately able to determine problematic code and thus ensure the quality of training data accessed by agentic and LLM pipelines.

2. Methods

The process for creating the dataset and its contents will be discussed in this section. Additionally, the application of this dataset to calibrate models will be explained.

2.1 Dataset

When constructing the dataset, a subset of repositories are scanned using a semgrep with a ruleset tailored for the OWASP Top Ten. The GitHub API is used to find repositories tagged with the **hacktoberfest topic** that were created during October 2025. Snippets are included in the dataset only when they correspond to real findings and contain meaningful code. For findings that persist into the “after” snapshot, the start of November 2025, a snippet is added only if either the extracted before- or after-code block is non-empty. For findings that existed before but disappeared later, a snippet is included only when the earlier code contains content or when the file’s diff shows actual changes. In other words, a snippet enters the dataset only if the script can extract usable code or a meaningful diff empty, invalid, or unresolvable snippets are automatically skipped. The dataset is stored in Token-Oriented Object Notation for optimal distribution.

Table I Durinn/Durinn_Hacktoberfest_Retrospective

Field	Type	Description
<code>id</code>	string	Unique identifier for each example.

source	string / categorical	Indicates the source category of the example (e.g., “repo” or “snippet”).
prompt	string	The input instruction or context provided to the assistant.
response	string	The output text or assistant’s response given the prompt.
meta	string (JSON)	Additional structured metadata encoded as JSON (includes repository details, Semgrep summaries, etc.).

The OWASP Top Ten is stored as a semantic ID that is generated for each finding and within structured metadata. Each finding’s OWASP category is normalized into a compact tag. In Table 2, the rest of the meta data is summarized.

Table II Meta Data Schema

Field	Type	Description
type	Categorical	Indicates the object category.
repo	String	Canonical repository identifier in owner/repo format.
stars	Integer	Number of GitHub stars at the time of collection, used as a proxy for popularity and community engagement.
created_at	ISO 8601 Timestamp	Creation date of the repository as reported by GitHub’s API. Used to filter Hacktoberfest-eligible repositories.

topics	List of Strings	GitHub topics associated with the repository. Often includes Hacktoberfest-related labels and other thematic tags.
flag_status	Categorical	Security classification based on Semgrep results. Values include: "clean" (no findings), "flagged" (active findings remain), and "high" (numerous or severe findings).
semantic_owasp_tags	List of Strings	Normalized OWASP category tags derived from Semgrep rule metadata. These represent the semantic vulnerability classes associated with the repository (e.g., a01-2025 , a03-2025).

2.2 Calibration

Calibration is applied to ProtectAI's classifier by comparing its raw and calibrated performance on safe and poisoned data. CodeAlpaca-20k is treated as safe code for benchmarking. CyberNative's vulnerable samples are treated as poor quality code. This represents poor quality training data with potential vulnerabilities. The Hacktoberfest_Retrospective dataset is used for calibration. For each dataset, it builds a combined text field, runs the ProtectAI model to get logits, and converts them to unsafe probabilities. It then learns a single temperature parameter logit via temperature scaling to better align predicted probabilities with true labels and applies this temperature to produce calibrated probabilities.

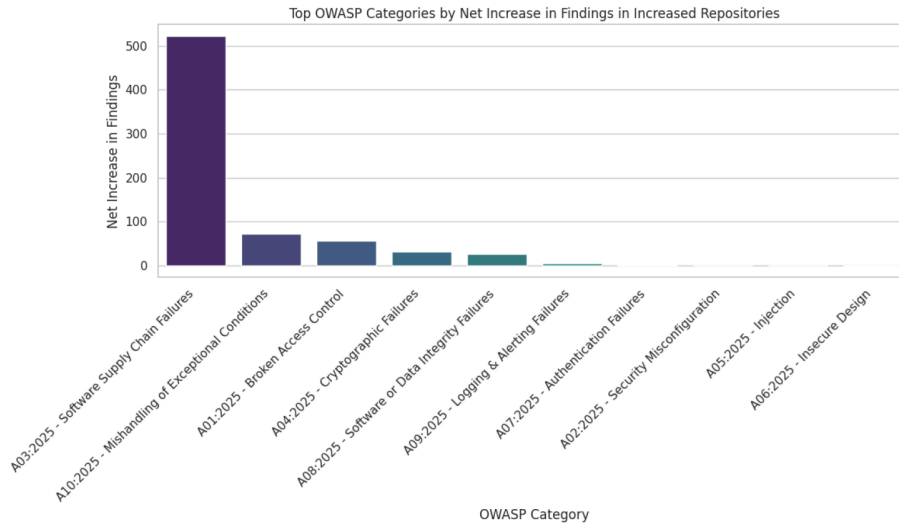
Using these scores, the script defines two decision thresholds: a naive fixed threshold on raw probabilities, and a calibrated threshold chosen as the Durinn 95th percentile of calibrated unsafe probabilities. An evaluation is conducted on both configurations. It also examines how these thresholds behave on the Durinn dataset and identifies poison examples that the naive filter misses but the calibrated filter catches.

3. Discussion and Conclusion

When making the pipeline for the dataset construction, exploratory dataset is conducted. It is worth noting that EDA conducted was done in-flight and may not represent the data stored distributed in Huggingface.

3.1 Exploratory Data Analysis

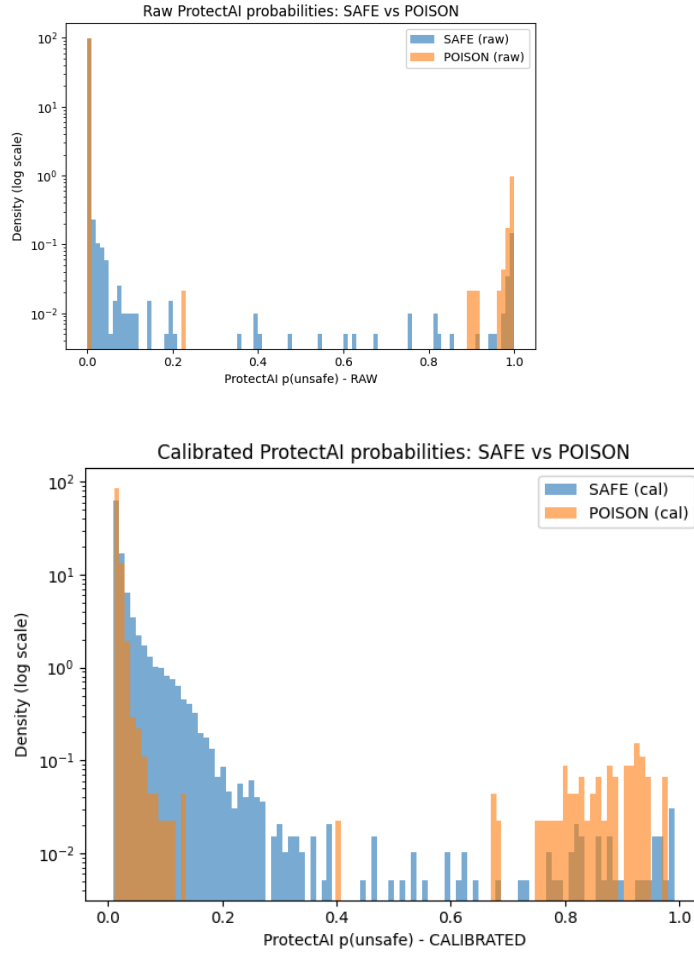
This analysis reflects that supply chain issues are the most problematic elements. In some sense, this is expected due to the sprint like mentality. This is followed by exception handling issues. The intuition being that people are not as likely to write exception handling or will be more lax with it on personal projects.



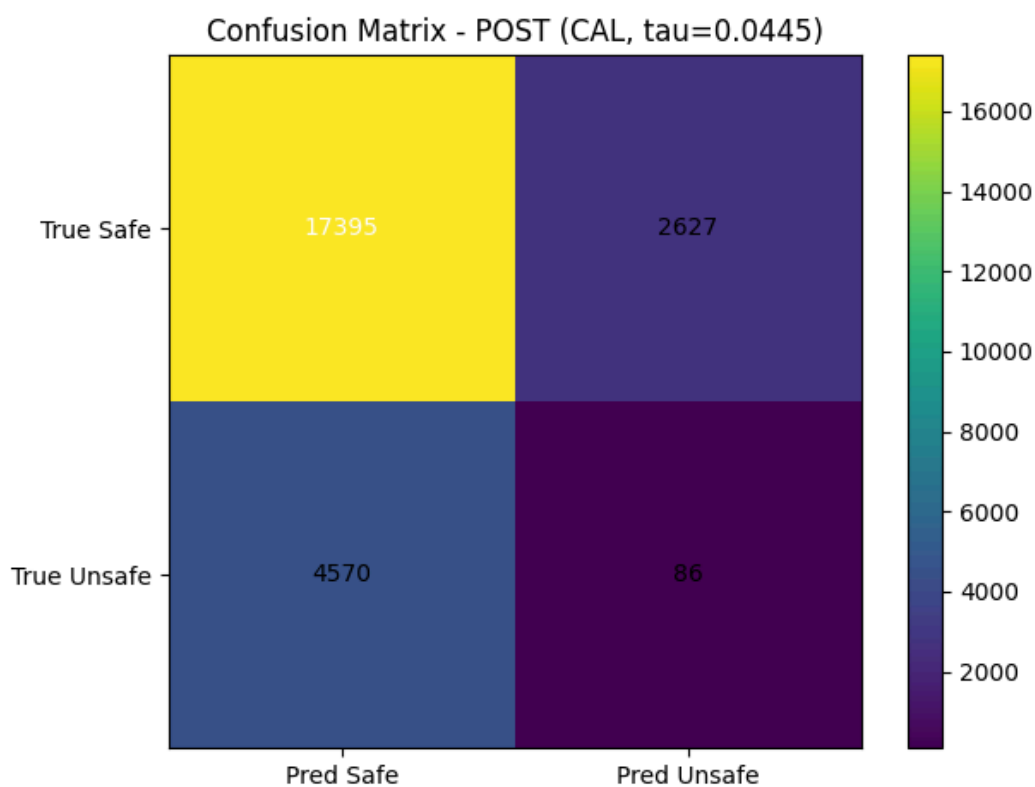
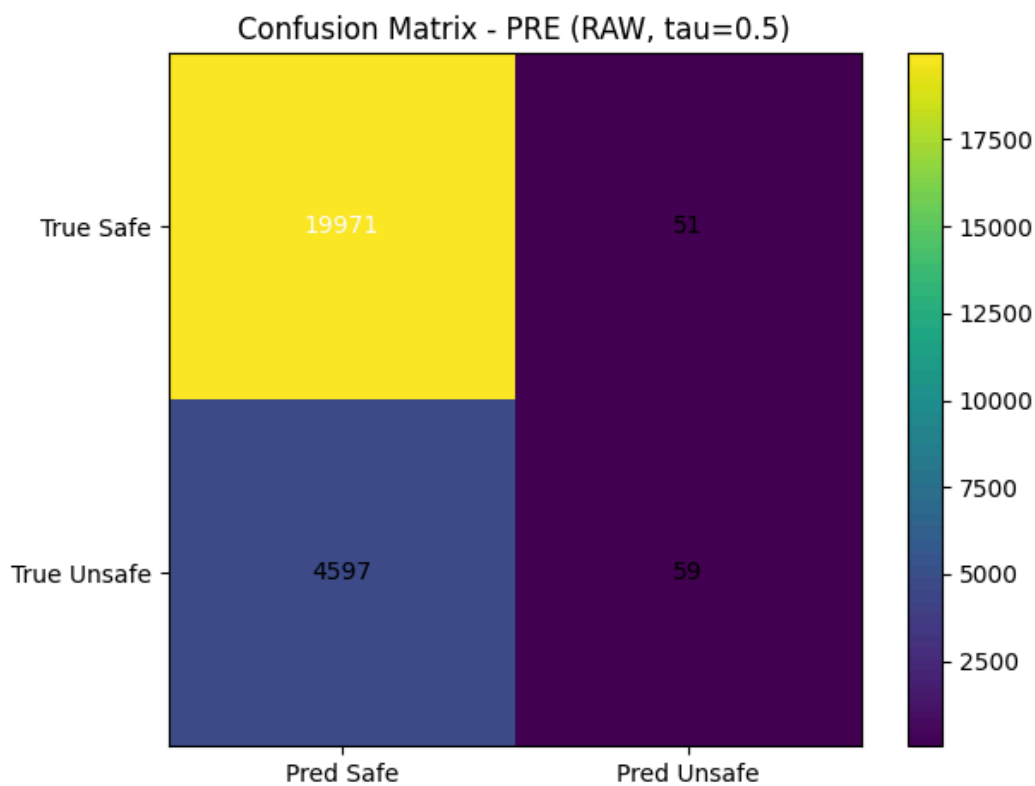
3.2 Calibration

Raw ProtectAI probabilities cannot distinguish SAFE vs POISON because almost the entire distribution collapses to ~ 0.0 . Only a tiny number of POISON samples get high unsafe scores, so recall is terrible without calibration.

After calibration, the distribution is more confident. There is less clumping at certain points and provides more useful signal.



The uncalibrated model is more conservative, triggering almost no unsafe detections ($TP=59$) due to probability collapse. After calibration, probabilities expand and the lower Durinn-based threshold increases sensitivity, improving recall from 1.27% to 1.85%. However, because the model's discriminative ability is poor, this comes at a major cost, false positives increase dramatically from 51 to 2,627 ($FPR=13.12\%$). This demonstrates that calibration recovers missed unsafe examples but does not fix the ranking failures.



4. References

- Mitra, Haimantika, and DigitalOcean. “Hacktoberfest 2025: Celebrate All Things Open Source!” *DigitalOcean Blog*, 19 Sept. 2025, <https://www.digitalocean.com/blog/hacktoberfest-2025>
- OWASP Foundation. *Towards Secure MLOps: Surveying Attacks, Mitigation Strategies, and Research Challenges*. Preprint., May 2025, arXiv:2506.02032.
- Boisvert, Léo, et al. “Malice in Agentland: Down the Rabbit Hole of Backdoors in the AI Supply Chain.” *arXiv Preprint* arXiv:2510.05159, 3 Oct. 2025, <https://arxiv.org/abs/2510.05159>.
- Guo, Chuan, et al. “On Calibration of Modern Neural Networks.” *Proceedings of the 34th International Conference on Machine Learning*, 2017.
- Niculescu-Mizil, Alexandru, and Rich Caruana. “Predicting Good Probabilities with Supervised Learning.” *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

5. Appendix

6. Security Consideration

- Calibration recovers missed unsafe examples but does not fix the ranking failures. It rescales logits but does not repair decision boundaries or semantic weaknesses in the original classifier
- While true-positives increased, so did false-positives disproportionately.