Enabling Live Behavioral Monitoring of Mission-Critical AI Systems

David Oygenblik
Email: davido@gatech.edu

The Department of Energy's National Renewable Energy Laboratory (NREL) increasingly leverages AI for mission-critical applications vital to national energy security [1], [2]. However, this reliance introduces significant security risks within NREL's critical energy infrastructure applications, caused by the ever expanding attack surface of AI (e.g., adversarial inputs [3], backdoors [4], code-based backdoors [5], prompt injections [6], and poisoning at scale [7]).

Consider an AI system used and developed by NREL to analyze real-time sensor data for predicting grid instability. An adversary has manipulated the model code [5] or its training data [4], [7], implanting a backdoor designed to activate under specific grid conditions (a scenario highlighted by my prior work [8]). Under the adversary-specified trigger conditions, the system may recommend control actions that exacerbate instability, leading to cascading failures across the power grid–a mistake with severe economic and public safety implications. An NREL scientist only monitoring physical world changes in isolation would be too late to detect such AI attacks, underscoring the urgent need for the AI model behavioral monitoring techniques in this proposal.

Multiple centers at NREL, such as Energy Security, Resilience and Integration (ESRI), Cybersecurity Research Center (CRC), and others [9], are already making strides in understanding and mitigating AI security threats within critical infrastructure. This proposal aims to directly complement and enhance these ongoing efforts at NREL. The proposed research will develop capabilities to analyze black-box AI systems used and developed by NREL to enable real time monitoring of AI system behavior.

Prior Work. Unfortunately, prior AI testing efforts [10], [11] assume idealized white-box conditions for proprietary models or ignore the AI system's encapsulating software used within complex workflows [12], [13]. Second, existing research only considers weight attacks [10], [14] but ignores code-based attacks [5], [15] entirely. Importantly, my own prior work, AiP [16], while successfully recovering the model's weights and topology, did not capture the model's code implementation, which is crucial for analyzing code-based threats and enabling reuse for models with custom-implemented operators. Finally, prior AI testing tools are one-shot, and are not run alongside the AI model for runtime anomaly detection (e.g., robustness to adversarial examples).

Objective. I propose AEGIS (**AI E**xamination and **G**uarding through **I**mplementation **S**crutinization), an automated pipeline for the analysis and monitoring of black-box AI systems. AEGIS delivers three key impacts (Fig 1):

Task I (Model Recovery): Task 1 will provide a novel methodology to recover a model's complete unified representation (weights/topology and its inference code) necessary to create an instrumentable, white-box model. This task will build upon my previously published model recovery and reuse methodologies [16], [17] enabling model vulnerability analysis (Task II) and real-time behavioral monitoring (Task III).

Task II (Security Evaluation): Before redeploying the AI system, AEGIS will apply novel behavioral profiling techniques that combine white-box methods with program analysis to identify weights and *data/control flows* [5], [8] in the model that are susceptible to minor perturbations to the input space (and will cause model misclassifications). Based on profiling, AEGIS will generate boundary constraints for model operation at runtime (Task III).

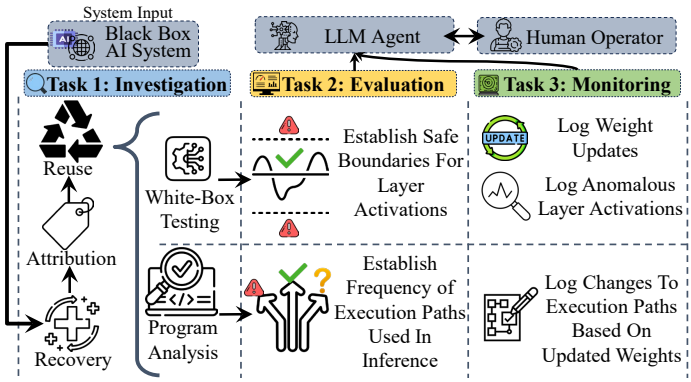

Figure 1: AEGIS Overview.

Task III (Monitoring): Then, AEGIS will instrument the model code with boundary constraints (from Task II) to identify anomalous behavior at runtime. AEGIS will employ an LLM that interfaces with human operators, and detects/summarizes

anomalous model behaviors for operators.

1 Task I: Enabling Forensic Investigation of Mission-Critical DOE-AI Systems

Preliminary Results. Task 1 builds upon my prior work (AiP [16] and ZEN [17]), which demonstrated recovery of diverse AI models (e.g., YoloV5-HIC [18], Llama 2-PT [19]) and their inference code from memory (Tab 1) for model reuse. Tab 1's results provide confidence in Task 1's success because a post-recovery performance identical to original deployment was achieved (Columns 5-6).

Table 1: Model Recovery on Customized Models [16], [17].

Model	AiP/ZEN Recovery			Post-Patch Reuse	
	Weights	Funcs	Base Model	Deployed	Recovered
YV5-HIC [18]	9.3M	341	YoloV5 [20]	91.1%	91.1%
YV8-Gold [21]	24.2M	750	YoloV8 [22]	93.9%	93.9%
RN-SE [23]	5.7M	8	Resnet [24]	92.3%	92.3%
MNV2-Ghost [25]	4.7M	39	MNV2 [26]	82.8%	82.8%
nGPT-LORA [27]	134.7M	29	nGPT [28]	96.8%	96.8%
Llama2-PT [19]	6.7B	20	Llama2 [29]	100%	100%

Overall Strategy. Task I will start from CPU/GPU memory snapshots of a black-box AI system (Fig 1, left column). AEGIS will first recover the model's weights, layers, and architecture from memory via the recovery of directed-acyclic-graph data structures (the way the model's topology is represented in code). Then, AEGIS will recover the code implementing the model's inference (implementation of each node in the model's topology). This builds upon my prior published works' (ZEN [17] and AiP [16]) capabilities to establish a unified representation of the model's recovered mathematical structure (topology) and model's layer implementations (code from memory). Taking the unified representation as input, AEGIS will identify an appropriate base model that can be instantiated and patched for testing (Column 4 in Tab 1). This methodology allows AEGIS to produce an instrumentable white-box instance of the deployed AI system, ready for behavioral profiling in Task II.

Experimental Design. In addition to ZEN's prior evaluation, AEGIS will be evaluated using a diverse testbed of AI models (spanning vision/language domains, including base and customized variants) *that are relevant to models used at NREL*. The experiment setup will compare recovered structural and code components against ground truth, identify a base model for a given model variant, and verify that the AEGIS-reconstructed model's performance matches the pre-recovery performance. Success metrics will include recovery accuracy, attribution accuracy, and model-reuse fidelity.

Potential Problems. Highly domain-specific models (such as those that may be used at NREL) may diverge significantly from known base models, or multiple base models might share substantial code/structure, potentially leading to ambiguous or incorrect attribution by AEGIS. This can be mitigated by initially focusing evaluation on models with clear provenance, then manually identifying lineage for significantly divergent models through reverse engineering to expand the base model library.

2 Task II: DOE-AI System Testing and Boundary Constraint Generation

Overall Strategy. With the unified representation in hand from Task I, Task II will develop novel methodologies to identify weight-based backdoors [4], architectural backdoors [5], and novel hybridized attacks (such as those illustrated in my prior work, VillainNet [8]), to then create constraints (center column of Fig 1) for monitoring model behavior at runtime in Task III. AEGIS will first generate code-level Control and Data Flow Graphs (CDFGs) to describe the model's control logic given data flow through its inference code (using benign test data), to establish behavioral signatures of benign model inference. Given behavioral signatures for benign test data, AEGIS aims to detect weights/control flows in the model that are sensitive to small input perturbations. Intuitively, components of the model (e.g., weights, control flows, etc.) that can be activated to consistently cause moderate to severe performance degradations (decreases in accuracy) at inference time should be monitored. By forcing activation of execution paths in inference code or forcing high activation of weights in the model, AEGIS gathers information as to where, and how, components of the model can be activated to degrade performance. AEGIS uses this information to inform the construction of behavioral constraints in Task III.

Experimental Design. AEGIS's Task II capabilities will be evaluated using models recovered in Task I. This testbed will include both benign models and variants in which there are backdoors (either weight-based, code-based or hybridized). The experiment setup will compare AEGIS's CDFG generation relative to ground truth CDFGs, compare AEGIS's identification of malicious layer activations/backdoored code from benign code, and measure AEGIS's overhead for behavioral profiling. Evaluation will assess AEGIS's

CDFG generation relative to ground truth CDFGs; overhead of code-aware execution path tracing and behavioral profiling; and the accuracy (e.g., precision, recall, F1-score) of AEGIS's anomaly detection to differentiate backdoored code, or malicious layer activations, from benign ones. Success metrics will include precision of CDFG generation, time in seconds of overhead, and precision for identification of malicious activations/code.

Potential Problems. Automatically differentiating backdoored code from complex, but benign, custom logic based solely on statically generated behavioral profiles, is challenging and may yield false positives or negatives. This can be resolved by combining dynamic execution profiles with static code features and developing heuristics based on known architectural backdoor signatures [5], [8].

3 Task III: Continuous Monitoring and Human Interfacing for DOE-AI Systems

Overall Strategy. AEGIS will employ a live monitoring framework, with an LLM-agent in the loop to interact with the human operator for any model behavioral abnormalities (right-most column of Fig 1). First, AEGIS instruments the inference code at each layer utilizing the behavior constraints generated in Task II to continuously monitor weight activations during deployment. Then, AEGIS will detect anomalies in layer activations that exceed the boundary conditions from Task II. Similarly, AEGIS identifies unexpected changes in code execution paths (e.g., early stopping [30]) which could trigger attacks on even seemingly benign inputs. AEGIS's LLM-Agent will summarize the layer activations and code execution paths, attribute anomalies to specific weight activations or execution path changes, and generate reports for human operators.

Experimental Design. AEGIS's Task 3 live monitoring capabilities will be evaluated using the models from Task II, subjected to simulated deployment scenarios relevant to NREL tasks. Evaluation will assess: (1) Monitoring fidelity, by verifying the accurate logging of control/data flows and relevant internal behaviors (e.g., weight activations); (2) Accuracy of anomaly identification (e.g. F1-score); (3) LLM Agent Effectiveness, via qualitative analysis measuring the LLM's ability to correctly interpret anomalies and accurately summarize behaviors attributed to specific updates.

Potential Problems. Mission-critical applications of DL models necessitate high inference speeds. Instrumentation of inference code with additional logging or measurement code can add significant overhead to performance time. To mitigate this, AEGIS introduces event-based logging into inference that only occurs after an anomalous event has occurred (e.g., misclassification). There is also a fundamental disconnect between anomalous weight activations/control flows and *how* the LLM agent will report this to the user. Bridging this gap, AEGIS will report system-level behavioral anomalies and corresponding mitigations to the user rather than details about the specific component that was the root cause of model misbehavior.

4 Expected Results and Timeline

Expected Results. This project will produce open-source repositories containing AEGIS's implementation and usage guide. My prior research, corresponding to Task I, has already produced two publications in top-tier cyber security venues **AiP**, [17] and research-community evaluated artifacts. The remaining two tasks of AEGIS are expected to also produce publications in top-tier cyber security conferences. In addition to academic publications, we plan to give real-world demos of the application of AEGIS to NREL relevant models and systems to NREL engineers.

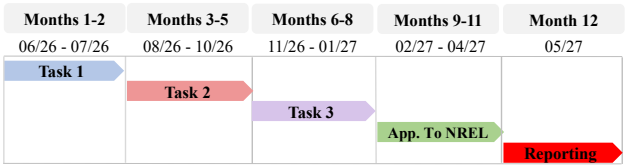

Figure 2: AEGIS Implementation Timeline.

Timeline. This 12-month research plan (shown in Fig 2) aligns project tasks with my doctoral thesis goals, utilizing NREL's Energy Security and Resilience Center resources. Months 1-2 will focus on applying my prior AI model code and attribution techniques to enable model reuse of NREL models (Task I). Months 3-5 will focus on security evaluation via runtime behavioral boundaries (Task II) in NREL-relevant scenarios. Months 6-8 will implement the live monitoring framework and LLM integration in NREL's simulated environments (Task III). Finally, months 9-12 will include application of AEGIS to real NREL systems and reporting of results to NREL engineers.

Bibliography

[1] National Renewable Energy Laboratory, *Advanced Research on Integrated Energy Systems*, https://www.nrel.gov/aries, [Online; accessed: 2025-04-10], 2025.

[2] National Renewable Energy Laboratory, *Security of Artificial Intelligence*, https://www.nrel.gov/computational-science/security-of-artificial-intelligence, [Online; accessed: 2025-04-10], 2025.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6572.

[4] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proc. 32nd NeurIPS*, Dec. 2018.

[5] H. Langford, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural neural backdoors from first principles," in *Proc. 46th IEEE Security and Privacy*, May 2025.

[6] J. Shi, Z. Yuan, Y. Liu, Y. Huang, P. Zhou, L. Sun, and N. Z. Gong, "Optimization-based prompt injection attack to llm-as-a-judge," ser. CCS24.

[7] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *Proc. 45th IEEE Security and Privacy*, May 2024.

[8] D. Oygenblik, A. Vemulapalli, A. Agrawal, D. Sanyal, A. Tumanov, and B. Saltaformaggio, "Villainnet: Targeted poisoning attacks against super networks along the accuracy-latency pareto frontier," in *Proc. 32nd ACM CCS*, Oct. 2025.

[9] National Renewable Energy Laboratory, *Energy Security & Resilience Research*, https://www.nrel.gov/security-resilience/, [Online; accessed: 2025-04-10], 2025.

[10] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. 40th IEEE Security and Privacy*, May 2019.

[11] DARPA, *GARD*, https://www.darpa.mil/research/programs/guaranteeing-ai-robustness-against-deception, [Online; accessed 2025-02-07], 2019.

[12] R. Wu, T. Kim, D. (Tian, A. Bianchi, and D. Xu, "DnD: A Cross-Architecture deep neural network decompiler," in *Proc. 31st USENIX Security*, Aug. 2022.

[13] Z. Liu, Y. Yuan, S. Wang, X. Xie, and L. Ma, "Decompiling x86 deep neural network executables," in *Proc. 32nd USENIX Security*, Aug. 2023.

[14] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. 26th ACM CCS*, Nov. 2019.

[15] M. Bober-Irizar, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural backdoors in neural networks," in *Proc. 40th IEEE/CVF CVPR*, Jun. 2023.

[16] D. Oygenblik, C. Yagemann, J. Zhang, A. Mastali, J. Park, and B. Saltaformaggio, "AI Psychiatry: Forensic Investigation of Deep Learning Networks in Memory Images," in *Proc. 33rd USENIX Security*, Aug. 2024.

[17] D. Oygenblik, D. Dermenzhiev, F. Sofias, M. Yao, H. Xu, R. Zhang, J. Park, A. Sikder, and B. Saltaformaggio, "Achieving ZEN: Combining Mathematical and Programmatic Deep Learning Model Representations for Attribution and Reuse."

[18] S. Tang, S. Zhang, and Y. Fang, "Hic-yolov5: Improved yolov5 for small object detection," *2309.16393*, 2023. [Online]. Available: https://arxiv.org/abs/2309.16393.

[19] F. Galatolo, *vanilla-llama*, 2023. [Online]. Available: https://github.com/galatolofederico/vanilla-llama.

[20] G. Jocher, *YOLOv5 by Ultralytics*, 2020. [Online]. Available: https://github.com/ultralytics/yolov5.

[21] C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang, "Gold-yolo: Efficient object detector via gather-and-distribute mechanism."

[22] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLO*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[23] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. 35th IEEE/CVF CVPR*, Jun. 2018.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 33rd IEEE/CVF CVPR*, Jun. 2016.

[25] C. Chen, Z. Guo, H. Zeng, P. Xiong, and J. Dong, "Repghost: A hardware-efficient ghost module via re-parameterization," *arXiv:2211.06088*, 2022. [Online]. Available: https://arxiv.org/abs/2211.06088.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. 35th IEEE/CVF CVPR*, Jun. 2018. DOI: 10.1109/CVPR.2018.00474.

[27] S. Hayou, N. Ghosh, and B. Yu, "Lora+: Efficient low rank adaptation of large models," 2024. eprint: 2402.12354. [Online]. Available: https://arxiv.org/abs/2402.12354.

[28] karpathy, *Nanogpt*, [Accessed: 2024-07-11]. [Online]. Available: https://github.com/karpathy/nanoGPT.

[29] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, and G. Cucurull, "Llama 2: Open foundation and fine-tuned chat models," *2307.09288*, 2023. [Online]. Available: https://arxiv.org/abs/2307.09288.

[30] H. Barad, J. Achterberg, T. P. Chou, and J. Yu, "Accelerated ai inference via dynamic execution methods," *arXiv:2411.00853*, 2024. [Online]. Available: https://arxiv.org/abs/2411.00853.