meshco
Real Hacking – Right Now!

April 1st, 2025
We're not Joking. Good EGGs are the SANE Approach

So What?

The promise of effective human machine interaction in securing cyber space is truly exciting. However, the safety, security and reliability of those interactions and dependencies is not guaranteed or easy. Trustworthy and Responsible Generative AI systems must be designed as such.

Our approach has demonstrated a practical way to create, interface with, deploy and operate new AI enabled & enhanced capabilities. Further, we have proven that this can be accomplished with commodity compute assets that can be easily acquired and not subject to availability, blockades. tariffs or other market constraints.

Simply put, AFIRM Hackbot™ can be trained in the cloud to operate in the field and at the edge while easily integrating into existing cyber operations.

Introduction

5G is an evolutionary step in supporting the next wave of IoT (Internet of Things) solutions and deployments. With needed enhancements in data rates, cybersecurity, and connection volume, there is a near-limitless number of new IoT applications on the horizon: from smart cities to precision agriculture; from military to manufacturing. However, these new applications also present their own challenges, there is often still too much data to effectively and efficiently transmit, store, process and protect.

Active Forensic Intelligent Response Method (AFIRM) is a methodology that values simplicity and leverages existing capabilities. Its no-nonsense approach to holistic safety & security is bolstered by field-proven tactics, techniques and procedures. All relevant system event signals are captured, catalogued, ingested, analyzed, and preserved based on agreed system parameters and mission objectives. We refer to this capability as "Total Telemetry". In conjunction with a Federated Data Fabric (FDF) architecture and defined operational principles, it is positioned to emerge as a standard for making information systems "Audit-Ready", quantitatively less vulnerable (Safer) to attack and truly resilient, even when system elements are isolated.

The utility of this approach can be realized within any computing environment. Transportability and scalability is ensured by creating various abstraction layers and adopting a "Software Defined Everything" approach.

Problem statement

Even before the media hype and market's obsession with ChatGPT, the transformational value and potential of Machine Learning (ML), Large Language Models (LLMs) and Generative Artificial Intelligence (AI) was obvious. As we acknowledge the nation-state AI "Space Race" we must grapple with the fall-out of "Free" AI for all. Cheaters, deep fakes, poisoned & biased models, hallucinations and more conspire to make the use of AI dangerous, possibly more so than the problems they purport to address. Trustworthy and Responsible AI capabilities will not naturally evolve in the current eco-systems (see NIST AI 600-1). They will need to be designed to operate as such. Dependencies on external resources introduce unacceptable risks and costs for many stakeholders, especially those operating at the edge or in the field. Our approach addresses these and other less obvious issues associated with current and emerging methods

Objectives and scope

The primary objective of this research project was to determine the parameters that have material impact on the function and performance of integrated purpose-built models, trained and tuned LLM's into an existing Secure Adaptive Network Environment (SANE). Further, the researchers wish to determine if AFIRM Hackbot™ development will prove the viability of using commodity laptops, networked GPUs, and shared RAM architectures to support an isolated Cyber Reasoning & Response System (CRRS).

Methodology & Approach

All design, deployment and testing activity was to took place on two similar but diverse High-Performance Computing (HPC) platforms. The first was the 5G Advanced Communications Laboratory HPC platform. The system is physically located in Indianapolis, Indiana. The second HPC platform is located in the Dayton Research Lab of Meshco, Inc.

As previously mentioned, all system components were engineered with various abstraction layers, designed to avoid vendor lock-in, incompatibility issues, commercial constraints and undocumented dependencies. Using the reference architecture below (see Figure. 1) two functional high-level architectures were approved.

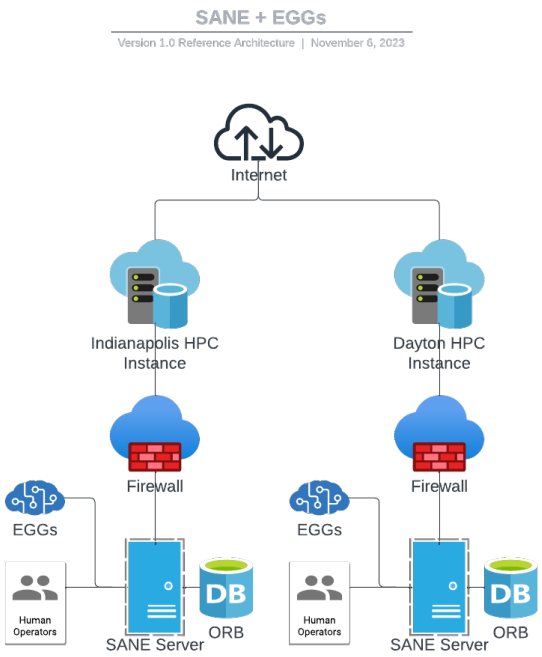


Figure 1

A third HPC platform was subsequently added. The Wright State University (WSU) HPC platform details can be found here.

Five foundational LLM's were considered for developing the initial rapid prototype. The five initial foundational models were selected based on several high-level eco-system characteristics; model license, model size, model popularity and diversity of use cases.

Models:

https://huggingface.co/NousResearch/Hermes-2-Pro-Llama-3-8B

https://huggingface.co/refuelai/Llama-3-Refueled*

https://huggingface.co/nvidia/Llama3-ChatQA-1.5-8B*

https://huggingface.co/tiiuae/falcon-7b-instruct*

https://huggingface.co/bigcode/starcoder2-15b

Further review and assessment of the initial five LLM candidates revealed three of interest: *Falcon7B, Llama3, and Nvidia's ChatQnA a Llama3 variant. Although the Falcon7B model could be utilized in its base form, training it proved to be an issue. Limited VRAM proved to be a bottleneck in fine-tuning models with high parameter counts. Initially the Dayton HPC platform was thought to have the lowest GPU resources. Thus, a requirement was set that all models selected to be compared during the project must be trainable on the Dayton HPC platform. Through extensive testing we discovered that models with parameters equivalent or less than four billion suited this requirement.

The three selected foundational LLM's were trained, tuned and tested in traditional high performance computing environments, shared computing environments and commodity hybrid system environments. A CI/CD pipeline workflow was developed to support automated containerization of models and workloads. The AFIRM containerized LLM's are referred to as Emergent Generative Generators (EGGs). The EGGs were put through a battery of tests to measure performance and functionality across the three reference compute platform environments. EGGs are purpose built to function in a specific role and labeled with relevant entity/EGG characteristics, such as foundational model provenance, training regime, temperature settings and operational costs. These characteristics and others are used to measure model performance, control EGG interaction and accomplish mission objectives. EGGs can be expressed as objects and are stored in the SANE (Secure Adaptive Network Environment) ORB (Object Repository Base).

Example EGG Object Descriptor:

NAME: Wayne **UID:** 001-PF-RAW-creation date/time stamp **Role/Class:** Path Finder **Level:** Oracle **Race:** Falcon7b **Experience Points:** TBD **Alignment:** Chaotic Neutral (no morality or ethics bias but with a slant toward non-traditional and multi-domain solutions. Radical thinking as a trait). **Backstory:** TBD **Skills:** OSINT workflow **Inventory/equipment:** crt.sh. shodan.io, whois, ping/tracert/icmp, breach db and Maltego

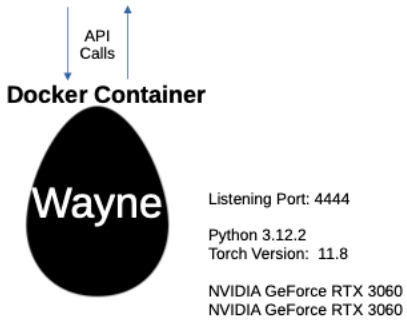


Figure 2

EGGs operate inside their own Docker container, so they can operate independently and in parallel. EGGs can communicate with other EGGs through restricted API calls that are logged as transactions. This allows for full control and telemetry of all system element interactions. (forensically sound). This abstraction layer allows EGGs to be “incubated” in one system environment and deployed & operated in a different system environment.

Dayton HPC Platform Medium-Level Design:

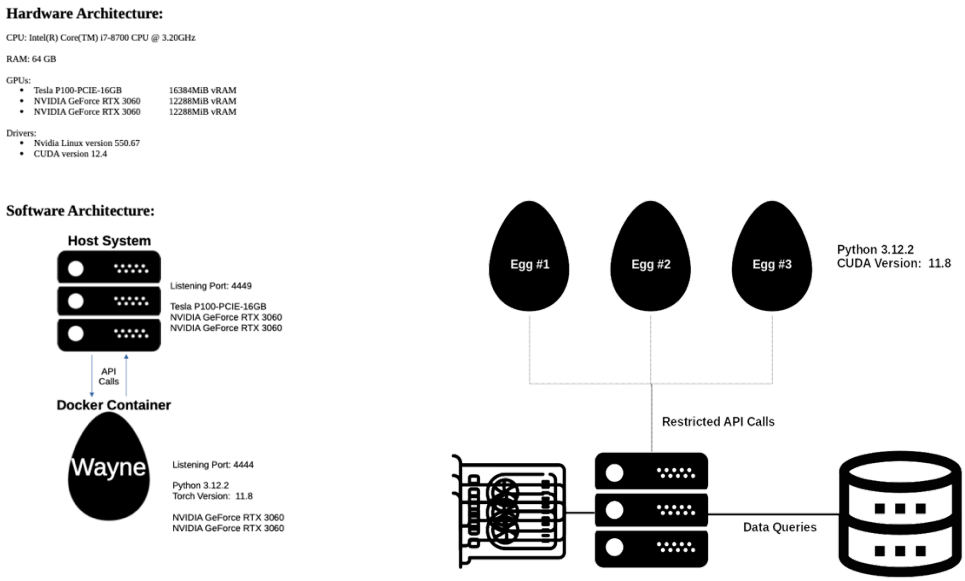


Figure 3

Model Scoring and Observations

The final model testing phase tested 12 models against the refined testing criteria:

https://huggingface.co/microsoft/Phi-3.5-mini-instruct

https://huggingface.co/togethercomputer/RedPajama-INCITE-Base-3B-v1

https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

https://huggingface.co/databricks/dolly-v2-3b

https://huggingface.co/openlm-research/open_llama_3b

https://huggingface.co/microsoft/phi-1_5

https://huggingface.co/microsoft/phi-2

https://huggingface.co/EleutherAI/pythia-1.4b

https://huggingface.co/EleutherAI/pythia-2.8b

https://huggingface.co/stabilityai/stablecode-completion-alpha-3b

https://huggingface.co/stabilityai/stablelm-3b-4e1t

https://huggingface.co/stabilityai/stablelm-base-alpha-3b

The base model of each of the twelve models were fine-tuned to produce two variants. The first variant specializes in answering general Cybersecurity questions and serves as a couch for junior pen-testers. The second variant specializes in providing python scripts to preform reconnaissance tasks and exploit vulnerabilities. Three models were selected to perform tasks using RAG: Llama3, Phi3.5, and Tiny llama. These models were selected due to their compatibility to the current RAG implementation process. Two embedders were utilized to generate vector databases from test pdfs for evaluation: bge-large-en-v1.5 and snowflake-arctic-embed-m.

Two curated datasets were used to teach models Cybersecurity knowledge and python coding. Two variants of each of the twelve models were produced. These model variants were trained through a single epoch due to time constraints. During training GPU power and VRAM usage were recorded to be used in performance evaluation of the variant models in their trained profession.

A hundred sample prompts were selected from each dataset and fed to the appropriate fine-tuned variants. Responses from variants for each prompt was scored against the expected response from 0 to 100. The average of all hundred prompts were used to determine the variant's overall accuracy. During the evaluation process other metrics such as time to generate response tokens, time for first response token, and max VRAM usage were logged.

To evaluate the Retrieval Augmented Generative (RAG) models, each of the models were given a sample set of one hundred prompts twice. In the first run, models were provided a vector database created using bge-large-en-v1.5. During the second run, models were provided a vector database created using snowflake-arctic-embed-m. Model responses were compared and scored against the expected outputs. Information on time to generate all response tokens, time for first response token, and max VRAM usage were logged during the evaluation process.

Through careful analysis of metrics gathered during the training and evaluation steps, three LLM models were selected as the optimal choice for each of the three tasks. These models were selected due to their high accuracy, compared to other tested models, while taking into consideration training time and resources used:

- Phyhia 2.8b was selected for tasks involving python scripting to preform reconnaissance tasks and exploit vulnerabilities.
- Phi 3.5 was selected for tasks involving general Cybersecurity questions and serves as a security co-pilot for junior pen-testers.
- Tinny Llama was selected alongside bge-large-en-v1.5 for tasks involving answering queries on information contained in a collection of pdf documents.

Phyhia 2.8b – Coding Variant

Platform	# of Test Sample Prompt	Seconds total (Mean)	Seconds total (Mean Std Dev)	Seconds to first token (Mean)	Seconds to first token (Mean Std Dev)	Tokens generated (Mean)	Tokens generated (Mean Std Dev)	Inference speed in tokens/sec (Mean)	Inference speed in tokens/sec (Mean Std Dev)	VRAM Memory (GB)
Dayton HPC	5.00	15.83	0.01	0.14	0.00	512.00	0.00	32.34	0.01	3.74
WSU HPC	5.00	65.51	6.11	3.43	0.38	476.80	0.00	7.74	0.47	6.41

Phi 3.5 – General Cybersecurity Variant

Platform	# of Test Sample Prompt	Seconds total (Mean)	Seconds total (Mean Std Dev)	Seconds to first token (Mean)	Seconds to first token (Mean Std Dev)	Tokens generated (Mean)	Tokens generated (Mean Std Dev)	Inference speed in tokens/sec (Mean)	Inference speed in tokens/sec (Mean Std Dev)	VRAM Memory (GB)
Dayton HPC	5.00	13.92	0.04	0.07	0.00	400.00	0.00	22.18	0.04	5.69
WSU HPC	5.00	55.49	0.32	0.90	0.22	381.60	0.00	11.65	0.05	8.51

Tinny Llama – RAG Implementation (bge-large-en-v1.5)

Platform	# of Test Sample Prompt	Seconds total (Mean)	Seconds total (Mean Std Dev)	Seconds to first token (Mean)	Seconds to first token (Mean Std Dev)	Tokens generated (Mean)	Tokens generated (Mean Std Dev)	Inference speed in tokens/sec (Mean)	Inference speed in tokens/sec (Mean Std Dev)	VRAM Memory (GB)
Dayton HPC	5.00	9.00	2.83	4.43	1.81	171.18	104.32	18.32	5.77	2.50
WSU HPC	5.00	7.67	3.15	3.97	2.67	152.31	66.95	21.49	7.62	2.50

The top three models that preformed best after a single epoch of training in general Cybersecurity knowledge are Phi-3.5-mini-instruct, stablelm-3b-4e1t, and open_llama_3b. These scored highest when evaluated on the new dataset they trained. The reason Phi 3.5 was chosen as the top model for the given tasks is because it out preformed all other models in accuracy by a large margin. It should also be noted that Phi 3.5 took the most time and resources to train out of all twelve models tested. Phi 3.5 was the costliest, but most accurate, of the twelve models when trained on general Cybersecurity knowledge.

Models	Training Time	Total kWh used	Estimated carbon emissions (lbs CO_2)	Estimated cost of electricity in Ohio ($)	Accuracy Mean Score (One Epoch Training)
Phi-3.5-mini-instruct	54081.28 s	1.9012	1.6351	0.30	50.53
stablelm-3b-4e1t	37458.94 s	1.3223	1.1372	0.21	42.28
open_llama_3b	47698.85 s	1.6904	1.4538	0.27	38.29

The four top models for tasks requiring python coding were Phi-3.5-mini-instruct, stablelm-3b-4e1t, pythia-1.4b, pythia-2.8b. Pythia-2.8b was selected as the top candidate at the time, based on its high accuracy compared to other models, while requiring less training time and resources. However, testing logs identified a power outage effecting 2 of the 12 model epochs. The effected models were put through their complete epochs. With the complete epochs in place, it was discovered that Pythia-1.4b out preformed Pythia-2.8b in accuracy after one epoch of training, while also drastically lowering the required training time and usage of resources.

Models	Training Time	Total kWh used	Estimated carbon emissions (lbs. CO_2)	Estimated cost of electricity in Ohio ($)	Accuracy Mean Score (One Epoch Training)
Phi-3.5-mini-instruct	10210.67 s	0.36	0.31	0.06	57.05
stablelm-3b-4e1t	7042.62 s	0.25	0.21	0.04	56.03
pythia-1.4b	3610.96 s	0.12	0.11	0.02	56.14
pythia-2.8b	7071.33 s	0.24	0.21	0.04	52.74

The embedder bge-large-en-v1.5 has the tendency to enable models to provide more accurate responses when compared against snowflake-arctic-embed-m.

Models	Vector Embedder	Accuracy Mean Score (With no Fine-tuning)
Phi-3.5-mini-instruct	bge-large-en-v1.5	59.75
Phi-3.5-mini-instruct	snowflake-arctic-embed-m	46.80
TinyLlama-1.1B-Chat-v1.0	bge-large-en-v1.5	50.75
TinyLlama-1.1B-Chat-v1.0	snowflake-arctic-embed-m	39.30
llama3	bge-large-en-v1.5	41.05
llama3	snowflake-arctic-embed-m	29.90

Conclusions

When comparing performance of selected models on the Dayton HPC and WSU HPC platforms, there was a noticeable difference in response speeds given the same prompts. Sample prompts for each task were tested five times to find the mean and standard deviations on each platform. The Dayton HPC platform proved to provide faster responses.

During testing of the fine-tuned models, it was noticed that all models scored less than seventy percent in accuracy. This is because models have not been fully fine-tuned through multiple epochs of training, and the subject (EGG Role/Class) being trained has a great influence over how models respond.

Some of the models analyzed were pre-trained to not provide instructions for preforming potentially illegal tasks. Part of the fine-tuning process was meant to circumvent these protections. With more training epochs, models will be able to provide more accurate responses and assist with tasks, in which they were previously taught not to assist.

Initial observations suggest a local purpose-built platform, such as the Dayton HPC platform, is the easiest to implement and most efficient to operate, even though the Dayton HPC platform didn't have the most computational resources. If the Dayton HPC platform is to be used for further product development it would benefit from more GPU resources to accommodate larger models.

Appendix A

How the Technology Works: AFIRM's Secure Adaptive Network Environment (SANE) architecture uses the Object Repository Base (ORB) to compare signals to Indicators of Compromise (IOC's), Tactics, Techniques and Procedures (TTP's), signatures, exploit primitives, enemy profiles, and other relevant referenceable objects necessary for the EGGs to determine and score possible responses including automated actions.

SANE implementations were originally designed and deployed to be used by human operators. Its automation functionality was accomplished with scripts and predefined logic, which can be triggered via temporal, conditional and manual cues. By deploying EGGs in SANE ecosystems, human operators can become more effective and responsive through the use of semi or fully autonomous capabilities. The flexible and extensible architecture provides abstraction layers, which allow EGGs to be trained, tuned, deployed on different compute platforms and easily integrated into existing ecosystems without disrupting current operations.

Currently, computer forensics add limited value to enterprise network environments. It is reactive and passive - a direct descendent of archaic criminal theory and law enforcement investigative practices. Its utility is only engaged after an intrusion or seizure has occurred. Often this is too late to help prevent a catastrophic event. This is unfortunate, considering the value such techniques could provide in overall network security. A more systematic and proactive approach is required.

The difficulty in securing systems connected to public, untrusted networks is compounded by the sheer volume of transactions on those networks. Users are constantly demanding new services and more access. Security and IT personnel are being asked to do more with less. The only way to maintain information superiority is to aggressively exploit multiple technologies. When vital systems are forced to utilize vulnerable technologies, contingencies for defending against those weaknesses must be built into the information infrastructure itself.

AFIRM is predicated on the assertion that, by effectively utilizing existing technology, a quantifiable level of acceptable risk can be achieved. However, the inverse is also true. Unless one can effectively utilize the technology they currently wield, no level of acceptable risk can be achieved.

Commercial products only satisfy part of the need. In order to effectively incorporate internally developed tools and off- the-shelf tools, we created a common user-interface, database, and infusion capability. Communications to data collection points and data control points are provided through secure channels. It is these components that are collectively known as Secure Adaptive Network Environment (S.A.N.E.).

Appendix B

References:

NIST AI 600-1 https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

AFIRM SANE https://www.afirm.org/sane_final.pdf?submitButtonName=Download%21

Wright State University HPC Details - https://github.com/wrightedu/Programmers-Guide-to-the-Galaxy/tree/master/Getting-Started/hpc

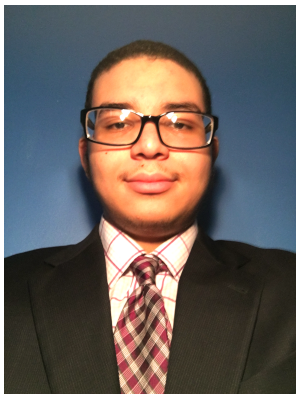
Appendix C

About the authors:

Bryan Fite

A committed security practitioner and serial entrepreneur, Bryan uses Facilitated Innovation to solve “Wicked Business Problems”. Having spent over 25 years in mission-critical environments, Bryan is uniquely qualified to advise organizations on what works and what doesn’t. Bryan has worked with organizations in every major vertical throughout the world and has established himself as a trusted advisor. “The challenges facing organizations today require a business reasonable approach to managing risk, trust and limited resources, while protecting what matters.”

https://www.linkedin.com/in/bryanfite/



Mohammed Kharij

Mohammed is a junior cyber security student at WSU. He studies cyber security from an adversary perspective in order to discover and patch overlooked flaws. One of his greatest goals is to contribute to the opensource community by developing and publishing tools.

https://www.linkedin.com/in/mohammedkharij/

