A Hybrid Machine Learning Approach for Red Team Log Analysis

Nickolas Mohr, Alan Shaffer, Gurminder Singh and Armon Barton
Naval Postgraduate School, Monterey, USA

nickolas.mohr@nps.edu
alan.shaffer@nps.edu
gsingh@nps.edu
armon.barton@nps.edu

Abstract: Red teaming is a common cybersecurity practice that simulates real-world adversarial cyber operations on defended systems to identify vulnerabilities. Current red team tools often have limited logging capabilities, resulting in insufficient analysis that prevents red teams from receiving real-time feedback and insights after operations. The application of machine learning for automating the analysis of red team operations is severely constrained by the scarcity of labeled, real-world log data. This research addresses this challenge by exploring the potential of using synthetic data to train attack-detection models for Cobalt Strike logs. We systematically evaluate three different training approaches for analyzing Cobalt Strike operational logs: synthetic-only, real-world-only, and a hybrid approach that combines both data types. Our methodology employs a comprehensive feature engineering pipeline that includes both programmatic log generation for creating large-scale structured data and large language model techniques for introducing variety and edge cases. We transform each log file into a high-dimensional vector that includes event types, command verbs, temporal activity patterns, and mappings to the MITRE ATT&CK knowledge base. Random Forest classification models are trained using this feature set to distinguish between successful and failed attack scenarios. By rigorously testing each training approach against a manually labeled ground-truth set of 112 authentic Cobalt Strike logs, we quantify the performance and limitations of each strategy. Our main contribution is demonstrating that a hybrid training strategy achieves 94% accuracy, greatly surpassing synthetic-only models (56%) and real-world-only models (79%). This combined approach effectively addresses both the domain gap in synthetic data and the data scarcity in small, real-world datasets. The hybrid model learns attack diversity from over 30,000 synthetic scenarios while grounding understanding in the authentic structural patterns of real logs, providing a 15 percentage-point improvement over real-data-only approaches. This research offers a practical framework for enhancing limited real-world cybersecurity datasets by strategically integrating synthetic data, enabling immediate use in Department of Defense red team operations and wider cybersecurity machine learning applications.

Keywords: Red team, Log analysis, Machine learning, Vulnerability assessment, Large language model

1. Introduction

In the modern cybersecurity arena, an organization is constantly under threat from malicious actors seeking to exploit its vulnerabilities. This digital arena is like a battlefield where attacks are not just possible but unavoidable. Therefore, it is critical for an organization like the U.S. military, which manages a huge amount of sensitive data, to prepare for potential incidents in cyberspace.

Red team operations are a crucial component of a proactive cybersecurity plan, as they simulate real-world attacks. Red teams help organizations identify weaknesses in their systems, processes, and staff before malicious actors can find and exploit them. The importance of red teaming cannot be overstated, especially in the realm of national security. As reviewed in the 2024 Department of Defense (DoD) Cyber Assessment Program report, the offensive capabilities of potential adversaries are escalating; many cyber defense and operations missions remain vulnerable to offensive cyber capabilities of potential adversaries (Director, Operational Test and Evaluation, 2024).

Red teaming has its roots in military strategy, where commanders tested battle plans by designating a group to play the role of the adversary. In the context of cybersecurity, the National Institute of Standards and Technology defines a red team as "a group of people authorized and organized to emulate a potential adversary's attack or exploitation capabilities against an enterprise's security posture" (National Institute for Standards and Technology, 2025).

While the importance of red teaming is widely recognized, little attention has been given to transforming their activity data, namely, log data captured in tools such as Cobalt Strike, into structured insights for analysis. These logs provide detailed information about operator actions, tools used, and mission outcomes; however, logs are often noisy and inconsistent. Another impediment is the significant lack of high-quality, labeled red team log data for analysis.

This research has addressed these problems using a hybrid synthetic data approach. We developed a programmatic log generator that produces labeled data at scale with consistent structure and accuracy. Large language model methods introduce variety and rare cases, but struggle with longer sequences and require careful quality control. Combining the LLM-generated logs with the programmatically generated logs establishes a training foundation that supports the development of accurate models and exposes them to unusual patterns. This research has demonstrated that the automated analysis of red team log data is feasible and that linking red team exercise data with machine learning can enhance insights into red team effectiveness.

2. Related Work

Red team tools generally fall into three categories: tools with advanced capabilities but devoid of logging features, those with limited or standardized logging capabilities, and those that provide extensive logging with some form of log analysis functionality.

Tools such as the Social Engineering Toolkit (SET) and Nmap are prime examples. These are open-source tools designed for easy use, but their focus on functionality comes at the expense of their inability to produce comprehensive user logs (Lyon, 1997; Kennedy, 2010).

The Metasploit Framework (MSF) is a popular open-source application for penetration testing, providing an extensive toolkit for employing and testing various payloads against targets. MSF offers a wide array of logging features, including logging a session, logging a database attack, and logging from within the MSF console (Moore, 2003). However, MSF cannot extract meaningful insights from its logs. It lacks data visualization, pattern matching, and automation features to help users identify successful actions, failures, or anomalies during penetration testing.

Similarly, PowerShell Empire offers red team operators a wide range of actions to take on a target host that has already been compromised. PowerShell Empire can log user actions, including agent check-ups, tasking, and executions, within the compromised system (Schroeder and Ross, 2015). The logs themselves offer a highly detailed record of the actions taken post-exploitation; however, like MSF, PowerShell Empire does not provide any pattern recognition or anomaly detection capabilities for logged data.

Log data is diverse, unstructured, and enormous in scale. It is generated by servers, devices, applications, operating systems, and networks, with each producing unique formats that can further complicate the data aggregation process (Raval and Verma, 2017)**.** Developers have created log management tools such as Splunk (Baum, Das, and Swan, 2003), Elastic Stack (Banon, 2010), and Graylog (Koopmann, 2010) to manage log data; however, these tools often struggle with processing diverse and unstructured logs (Novikov, 2017).

The Elastic Stack is an open-source solution that appeals to organizations for log storage, management, and analysis. However, the appeal quickly diminishes when data volume and complexity increase, requiring significant processing power to maintain performance (Banon, 2010). On the enterprise side, Splunk has established itself as the industry leader, particularly excelling in security analytics and IT operations. It enables users to view real-time analytics and insights from ingested log data. However, current licensing fees for Splunk can amount to $20,000 annually, not including the implementation of specialized analysis tools (Baum, Das, and Swan, 2003).

High false positive rates in log analysis tools lead to "alert fatigue," where analysts overlook genuine threats due to overwhelming volumes of false alerts (Tariq *et al.*, 2025). Advanced features, such as machine learning and anomaly detection, can help reduce false positives; however, not all tools have these capabilities built in. Even when available, such features often require significant additional setup and domain expertise.

Techniques such as supervised learning (decision trees, support vector machines), unsupervised learning (clustering), and reinforcement learning have proven effective in various cybersecurity applications, including anomaly detection and intrusion prevention (Buczak and Guven, 2015). The application of machine learning techniques to cybersecurity log analysis faces a fundamental challenge: the scarcity of high-quality, labelled training datasets. This problem is particularly acute in specialized domains, such as red team operations, where data is often classified or operationally sensitive. Traditional supervised learning approaches require substantial datasets of labelled examples to achieve reliable performance, creating a significant barrier to developing automated analysis capabilities for red team exercises.

3. Methodology

This section presents the hybrid synthetic data approach developed for automated red team log analysis. We describe the three data generation methods, feature engineering pipeline, and machine learning implementation used to classify operational outcomes from Cobalt Strike logs.

3.1 Overview and Design Goals

The core motivation for this research stems from the enormous challenges inherent in manually processing Cobalt Strike logs generated during red team operations. These logs, although rich in valuable information, are often dense, unstructured, and inconsistent in format, making manual review extremely time-consuming and prone to errors for operators and analysts. Traditional machine learning approaches face a fundamental barrier: the scarcity of labeled training data due to classification restrictions and operational security concerns.

This research addresses these challenges through a hybrid synthetic data approach that combines three distinct data sources: programmatically generated logs for scalability and consistency, LLM-based generation for diversity and edge cases, and authentic operational logs for grounding in real-world patterns. The methodology focuses specifically on Cobalt Strike logs, developing a comprehensive feature engineering pipeline that transforms raw log entries into structured, machine learning-ready datasets.

The primary scope of this work is to parse and analyze log files generated by the Cobalt Strike command and control framework. Our analyses focused on Cobalt Strike due to its widespread use within the U.S. military, and its well-documented format structure. The pipeline parses logs, links events to MITRE ATT&CK techniques, and uses Random Forests to classify operation outcomes, comparing synthetic-only, real-world-only, and hybrid training approaches to demonstrate the effectiveness of the combined methodology.

3.2 Data Generation Approaches

To address the fundamental challenge of data scarcity in red team log analysis, this research employs a three-pronged approach to data generation. Each approach contributes distinct advantages: programmatic generation provides scalability and structural consistency, LLM-based methods introduce tactical diversity and edge cases, and real-world logs ground the models in authentic operational patterns. The combination of these approaches enables comprehensive model training while maintaining the ability to validate performance against genuine red team data.

3.2.1 Programmatic log generation

A custom Python program named log_gen.py was created to generate synthetic logs that resemble those produced by Cobalt Strike. The generator utilizes randomized parameters, including beacon sessions, beacon IDs, IP addresses, and commands, to enhance realism and variety. Each log entry clearly shows whether it pertains to beacon initialization, tasking, or output, making precise and automated tracking easier during analysis. The log_gen.py file can be configured to generate dozens, hundreds, or even thousands of synthetic logs, each with a definitive ground-truth label indicating whether the simulated attack was successful.

The log sessions generated by log_gen.py use randomization and generate various MITRE ATT&CK techniques, including lateral movement, credential theft, and privilege escalation. Sequenced commands and their corresponding outputs identify successful attack chains, while failure chains are characterized by syntax errors, permission denials, or "access-denied" messages. Each synthetic log session is configured to randomly represent varying operator identities (such as "trainee1," "PenTester," "AlphaOp"), infrastructure components (including "DOMAIN," "ACME"), hostnames (e.g., "DC01," "Laptop-User1"), services (such as HTTPS, DNS, SMB), and ports (for example, 443, 80, 53). This approach allows the batch runner to efficiently automate the generation of thousands of log entries with balanced labeling.

3.2.2 Large language model-based generation

While the programmatic generator provided a scalable foundation of structurally consistent data, a second method was employed to enhance the dataset's diversity and introduce less predictable attack narratives. We generated some 150 additional logs using OpenAI's ChatGPT-4, as well as Claude and Gemini 2.5, following a controlled implementation process designed to ensure the quality and realism of the synthetic logs. The implementation of this LLM-powered generation began with detailed prompt engineering to guide the models' output. Rather than using generic requests, prompts were carefully crafted to provide the models with specific context, constraints, and a desired format based on the sanitized, real-world log examples in our possession.

3.2.3 Real world ground truth dataset

The foundation for all model validation and the anchor for our hybrid training approach was a set of 112 authentic Cobalt Strike logs derived from a real red team exercise. This dataset, referred to as FullLogAnalyzed, served as the definitive ground-truth for this research. Each log in this batch was manually analysed to determine the ultimate operational outcome of the red team operator's actions. This manual analysis generated a labelled dataset comprising 58 logs representing failed attack scenarios and 54 logs representing successful scenarios. While limited in size compared to the synthetic datasets, this batch was invaluable. It provided the authentic structural patterns, rhythms, and beacon life cycle events that our initial analysis had proven were missing from the synthetically generated logs, making it an essential component for both model validation and successfully bridging the synthetic-to-real gap.

3.3 Feature Engineering

The conversion of raw log data into predictive analytics was facilitated by a range of Python modules, each with a distinct function in the workflow, as illustrated in Figure 1. The analytical pipeline begins with preprocessor.py, which plays a crucial role in preprocessing and feature discovery. This first step is performed once to scan the entire collection of available log files, including both synthetic and real-world variations. Its goal is to identify and gather all unique features within the complete dataset. Features include various event types (e.g., beacon_checkin), operator names, effective command verbs (e.g., getsystem), and MITRE ATT&CK techniques. The output of this script is a JSON file called feature_metadata.json. This file acts as a master blueprint and comprehensive list of all potential features across the dataset. By establishing this data, it ensures that every feature vector generated in subsequent steps maintains a consistent structure.

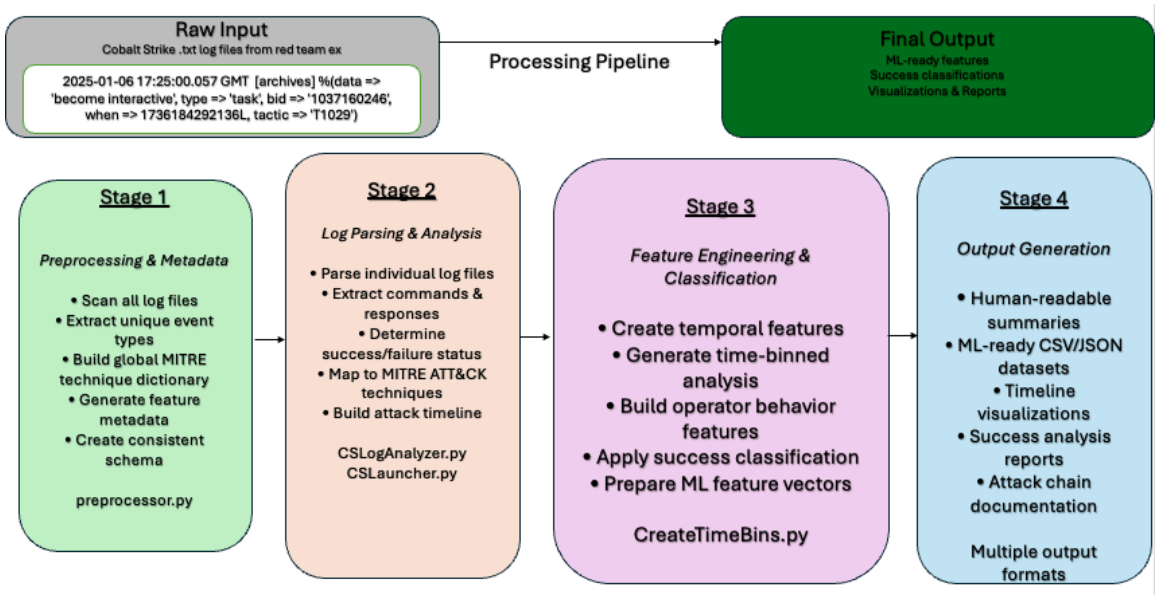

Figure 1: High-Level Architecture of the Cobalt Strike Log Analysis Tool

The log parser's primary role is to accurately calculate the number of these known features present in a specific log. It processes raw log entries and transforms them into structured data for analysis and ML feature extraction. The parser is specifically designed to handle Cobalt Strike logs, processing several key elements from each log entry: timestamps extracted in format "2025-07-01 03:14:07," event types categorized as "beacon_taskout" or "beacon_output," specific commands like "jump 10.0.2.8" or "hashdump," beacon IDs as unique identifiers for each session, operator information, source and target IPs extracted from network operations, session context derived from beacon metadata and prior activity, and success or failure labels determined from filename.

A key part of the data preparation process involves enrichment and labeling through MITRE ATT&CK technique mapping. The system incorporates MITRE ATT&CK technique labels through the Mitre_Mapping.py file, which contains a Python dictionary (MITRE_LOOKUP) that associates Cobalt Strike command verbs with corresponding MITRE ATT&CK techniques by their unique IDs and names. This structure enables a single Cobalt Strike verb to map to multiple techniques, providing standardized categorization of observed adversarial actions.

The final ML feature vector is structured to include the source log filename, one-hot encoded event types and MITRE techniques, log-level operator features such as binned start hour and total duration, and one-hot encoded time bin activity across 288 five-minute intervals over a 24-hour cycle. In this research, a total of 781 unique features were identified, creating comprehensive feature vectors suitable for machine learning analysis.

3.4 Machine Learning Implementation

With the data processing pipeline established, the final stage of implementation focused on the ML components. This involved two key phases: generating a comprehensive, ML-ready feature vector from the structured log data and selecting and implementing the algorithm used for training and validation.

For this research, the random forest classifier was selected as the primary ML algorithm. We chose this data classification model for several key reasons. It consistently performs well on structured data, such as the sets generated in our pipeline. It is also robust against overfitting, which is a significant risk when working with a high-dimensional feature space like our 781 features. Most importantly, a Random Forest offers a high level of interpretability through its feature importance scores, which highlight the features the model finds most predictive.

To conduct comprehensive testing, three distinct datasets were created and used in various combinations. The first was a Real-World Ground-Truth Dataset (FullLogAnalyzed), an authentic collection of 112 manually labeled logs from real-world red team operations. The second was an LLM-generated synthetic dataset (FullLogGpt), containing 123 logs with different and creative attack scenarios produced by LLMs. The final and largest source was a Programmatic Synthetic Dataset (LargeScaleTestData), which included over 30,000 logs generated by the Python script to ensure both massive scale and consistent scenarios.

To ensure a fair comparison between models, a single, consistent holdout set was used for final validation; the Real-World Ground-Truth Dataset was reserved for this purpose. Regardless of what data a model was trained on (synthetic, real, or a hybrid), its ultimate performance was judged based on its accuracy on real-world data, providing an unbiased comparison for every test case.

4. Results and Analysis

Our testing compared the performance of seven different models to answer three key questions. First, how well does a model trained on synthetic data perform on real data? Second, what is the best performance we can achieve using only a small set of real data? Finally, can a 'hybrid' model, trained on a blend of both data types, overcome the limitations of the other approaches to achieve the highest accuracy?

A central challenge in this research was the "synthetic-to-real" domain gap, which refers to the difference between the data patterns of a synthetic training dataset (the source) and a real-world dataset (the target). An ML model is said to generalize if it can accurately make predictions on new, unseen data that is different from the data it was trained on. If the model only memorizes its training data without learning any of the underlying issues, it fails to generalize.

Our initial tests were designed to measure this effect. The first model was trained exclusively on the LLM-generated synthetic data. While it achieved a near-perfect accuracy on its own test data (97% accuracy), its performance on the real-world holdout set collapsed to 48% accuracy. The model failed to generalize, as it learned and applied incorrect rules from the synthetic data to the real-world data. It was unable to identify a single failed attack in the real-world dataset correctly.

A second test model trained solely on synthetic data was developed using a much larger programmatic dataset. Its performance on real-world data slightly improved to 56%, but it still failed to generalize reliably, performing marginally better than a coin flip. A final model trained on a combination of both synthetic sources also underperformed, reaching only 48% accuracy. These experiments clearly show a significant domain gap, indicating that even a large and diverse collection of purely synthetic logs lacks the necessary patterns for a model to learn the correct rules and effectively generalize to real-world operational data.

To establish a performance baseline using only authentic data, a model was trained exclusively on the 112 manually labeled real-world logs. This model performed significantly better than the synthetic-only test cases, achieving an accuracy of 79%. This result is significant because it demonstrates that real-world logs contain consistent, learnable patterns that are fundamentally different from those in synthetic datasets. Additionally,

it highlights the issue of data scarcity. While 79% is a much better performance than seen with the synthetic models, the small number of training data ultimately limits the model's performance.

The hybrid approach proved to be a great success, significantly surpassing the baseline models. The Programmatic-Hybrid model achieved a peak performance of 94% accuracy. Its high precision (0.93-0.95) and recall (0.93-0.94) across both classes confirm that it learned to generalize effectively. This demonstrates that combining a large-scale, consistent synthetic foundation with a small authentic dataset is the optimal strategy for building a robust and reliable model.

4.1 Performance Summary

The comparison of the seven training models revealed interesting results and a clear order of performance. The experiments not only provided insights into the specific challenges of the synthetic-to-real domain gap and data scarcity but also confirmed the hybrid data approach as the best solution. As illustrated on Table 1, the Programmatic-Hybrid model, which combined large-scale synthetic data with a high-fidelity real-world dataset, emerged as the optimum model.

Models trained solely on real data reach a baseline accuracy of 79%, while integrating synthetic and real logs boosts accuracy to 94%. This demonstrates that programmatic generation provides scalability, LLMs introduce diversity, and real logs ground the model in authentic patterns. The approach also paves the way for future work in operator profiling through multi-class classification, as well as exploring other generative models like autoencoders and generative adversarial networks and applying the hybrid method to other fields with limited data.

The primary objective of this research was to develop and validate an effective ML pipeline for classifying the outcomes of red team operations based on log data. The comparison of three different training strategies revealed a performance hierarchy, offering clear insights into the challenges and solutions involved in using synthetic data for this task. The experiments began by establishing a baseline for a model trained exclusively on synthetic data, which achieved near-perfect accuracy (97%) on its synthetic test data but performance significantly declined when evaluated against a holdout set of 112 real-world logs, reaching only 56% accuracy. This confirmed the existence of a synthetic-to-real domain gap.

To establish a second baseline, a "real-only" model was trained exclusively on the 112 manually labeled real-world logs, achieving a respectable 79% accuracy. This result underscored the critical importance of authentic data patterns while simultaneously highlighting a performance ceiling imposed by data scarcity. The final experiment validated the "Hybrid Model" as the best solution, achieving an outstanding 94% accuracy on the same pure real-world validation set. This result represents a 38 percentage-point improvement over the best synthetic-only model and a 15 percentage-point improvement over the real-only baseline, demonstrating that the hybrid approach effectively addresses both the domain gap and data scarcity problems simultaneously.

Table 1: Comparative Accuracy of Training Methodologies on the Real-World Holdout Dataset

Training Method	Training Data	Data Ratio	Accuracy on Real-World Data
LLM-Synthetic Only	LLM-Generated	100% LLM	48%
Synthetic-Blend	LLM-Generated + Programmatic	1% LLM / 99% Programmatic	48%
Programmatic-Synthetic Only	Programmatic	100% Programmatic	56%
Real-World Only	Real-World	100% Real-World	79%
LLM-Hybrid	LLM-Generated + Real-World	57% LLM / 43 % Real	93%
Ultimate-Hybrid	Programmatic + LLM Generated + Real-World	98 % programmatic / 1 % LLM / 1 % Real World	93%
Programmatic-Hybrid	Programmatic + Real-World	99 % Programmatic / 1% Real World	94%

5. Discussion and Future Work

This research has demonstrated how the analysis of red team operations can be enhanced by creating and testing an ML pipeline that addresses log data scarcity, a key issue in the cyber domain. The primary challenge in this work was to determine if a model's predictive accuracy could be significantly enhanced by incorporating a substantial amount of synthetic data into a small, high-quality anchor dataset. By comparing seven different data set models, we developed an effective plan for overcoming the problem of data scarcity.

This research demonstrated that a small set of real-world log data (which alone can train a model to only 79% baseline accuracy) can be significantly improved by combining this data with a large synthetic dataset, to form a new Programmatic-Hybrid data model. When tested against the complete set of 112 real-world logs, this Hybrid Model achieved an impressive 94% accuracy rate. This demonstrates that real-world data instructs the model on what to look for, while the large synthetic dataset enables it to learn how to generalize that knowledge across thousands of variations.

This discovery shows that developing high-performing ML models does not require a large (and often hard-to-collect) set of real-world data samples. Instead, by using an approach that combines quality synthetic data with real-world operational log data, it becomes possible to create more accurate and effective data analysis models.

5.1 Implications for the Department of Defense

The insights gained from this research were transformed into actionable intelligence by developing an ML model that learns the signatures of a successful attack. This approach goes beyond simple log parsing and advances into predictive analysis. The model identifies actions among features, revealing patterns that may indicate a successful operation, such as a high frequency of specific data exfiltration techniques combined with a low frequency of error events. Once trained on the hybrid dataset, this model becomes a reusable tool that can be applied to future, unseen real-world logs to deliver data insights.

5.2 Future Research Direction

The primary approach of this thesis was to address log data scarcity by utilizing large-scale synthetic data anchored to a small, high-quality set of real-world data, which was shown to be highly adaptable. Future research should focus on applying this framework to new analysis areas and exploring more advanced generative techniques.

A significant impact of this research will be its potential application beyond cybersecurity. Many critical ML fields, particularly in the medical domain, are limited by the scarcity of large, labeled datasets due to privacy concerns and restrictions. A key area for future research is to perform experiments to validate this hybrid methodology in these other areas.

The model in this thesis was trained as a binary classifier to distinguish between success and failure. A logical next step is to develop a multi-class classifier for operator profiling. By training the model on logs labeled by individual operators, the pipeline could learn to identify each user's unique "signature" (such as their preferred tools, common errors, and operational tempo).

An interesting next step would be to adapt the log parser and log generation programs to support other common red team tools and frameworks, such as Metasploit and PowerShell Empire. This would confirm the validity of the hybrid training method across various log formats and help develop a more universal tool for automated red team analysis.

6. Conclusions

This research has demonstrated that the automated analysis of red team log data is feasible and that linking red team exercise data with machine learning can enhance insights into red team effectiveness. The hybrid synthetic data approach developed addresses the fundamental challenge of data scarcity in cybersecurity machine learning applications while maintaining the ability to achieve high performance on real-world operational data.

The key findings demonstrate that models trained solely on real data reach a baseline accuracy of 79%, while integrating synthetic and real logs boosts accuracy to 94%. This 15 percentage-point improvement represents a significant advancement in automated red team log analysis capabilities. The research validates that programmatic generation provides scalability, LLMs introduce diversity, and real logs ground the model in

authentic patterns, creating a comprehensive training methodology that overcomes both domain gap and data scarcity limitations.

The approach provides a practical framework for enhancing limited real-world cybersecurity datasets by strategically integrating synthetic data, enabling immediate use in Department of Defense red team operations and wider cybersecurity machine learning applications. By demonstrating that high-performing ML models can be developed without requiring large collections of classified operational data, this research opens new possibilities for advancing cybersecurity analytics while maintaining operational security requirements.

Ethics Declaration: This research involved analyzing cybersecurity log data from red team exercises. Ethical clearance was not required for this research as it utilized anonymized log files from existing Department of Defense training exercises with all personally identifiable information removed.

AI Declaration: The authors used generative AI to support multiple aspects of this research, including code development, idea generation, synthetic data creation, and writing improvement. Claude 3.5 Sonnet, ChatGPT 4, and Gemini 2.5 were used for several purposes: (1) to help debug and generate code for the log analysis tool, (2) to brainstorm ideas and concepts during the initial research process, and (3) to help generate synthetic Cobalt Strike logs that were used to train ML models. Grammarly was also used to proofread drafts and improve grammar, punctuation, and sentence structure during the writing process.

For code development, AI tools were provided with code functions and error messages, with suggested revisions implemented and tested using an iterative process. For idea generation, conversational interactions were used to explore research concepts and refine the approach. For synthetic log generation, AI was used to help create realistic training data that mimicked authentic Cobalt Strike log formats and scenarios. For writing, each grammar and style recommendation was considered individually.

To mitigate risks, control was maintained over all final decisions. All code implementations were thoroughly tested, synthetic data was verified to represent intended scenarios accurately, and all writing suggestions were reviewed before adoption. The analysis, conclusions, and technical implementation represent the authors' own work, with AI serving as a supportive tool rather than a primary source of content.

References

Banon (2010) “Elk Stack.” Amsterdam, NL. 2012. *Elastic Stack*, ver. 9.0.4. Available at: https://www.elastic.co/guide/index.html.

Baum, Das, and Swan (2003) “Splunk.” San Jose, CA, USA. 2003. *Splunk Machine Learning Toolkit*, ver. 9.4. Available at: https://help.splunk.com/en.

Buczak and Guven (2015) “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” *IEEE Communications Surveys & Tutorials*, 18(2), pp. 1153–1176. Available at: https://doi.org/10.1109/COMST.2015.2494502.

Director, Operational Test and Evaluation (2024) *Cyber Assessment Program*. Washington, DC, USA. Available at: https://www.dote.osd.mil/Portals/97/pub/reports/FY2024/dotemanaged/2024cap.pdf?ver=CYXAHEh1-0JMR-9Jde33wA%3D%3D.

Kennedy (2010) “Social Engineering Toolkit.” Fairlawn, OH, USA. 2010. *Social Engineering Toolkit*, ver. 8.0.3. Available at: https://github.com/trustedsec/social-engineer-toolkit.

Koopmann (2010) “Graylog.” Houston, TX, USA. 2009. *Graylog*, ver. 6.6. Available at: https://graylog.org.

Lyon (1997) “Network Mapper.” San Jose, CA, USA. 1997. *Network Mapper*, ver. 7.97. Available at: https://svn.nmap.org/!svn/bc/3320/nmap/docs/nmap_manpage.html≠.

Moore (2003) “Metasploit.” Boston, MA, USA. 2003. *Metasploit*, ver. 4.22.8. Available at: https://www.metasploit.com.

National Institute for Standards and Technology (2025) *Security and Privacy Controls for Information Systems and Organizations*. Rep. 800-53, rev. 5. Gaithersburg, MD, USA. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.

Novikov (2017) *Log Analysis*, *Wallarm*. Available at: https://www.wallarm.com/what/log-analysis.

Raval, V. and Verma, S. (2017) “Challenges of Security Log Management,” *ISACA Journal*, 6. Available at: https://www.isaca.org/resources/isaca-journal/issues/2017/volume-6/the-practical-aspect-challenges-of-security-log-management.

Schroeder and Ross (2015) “PowerShell Empire.” Las Vegas, NV, USA. 2010. *Powershell Empire*, ver. 6.1.3. Available at: https://github.com/BC-SECURITY/Empire.

Tariq *et al.* (2025) “Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities,” *ACM Computing Surveys*, 57(9), pp. 1–38. Available at: https://doi.org/10.1145/3723158.