Leveraging Sharing Communities to Achieve Federated Learning for Cybersecurity

Frank W. Bentrem†, Michael A. Corsello†, and Joshua J. Palm†

Abstract. Automated cyber threat detection in computer networks is a major challenge in cybersecurity. The cyber domain has inherent challenges that make traditional machine learning techniques problematic, specifically the need to learn continually evolving attacks through global collaboration while maintaining data privacy, and the varying resources available to network owners. We present a scheme to mitigate these difficulties through an architectural approach using community model sharing with a streaming analytic pipeline. Our streaming approach trains models incrementally as each log record is processed, thereby adjusting to concept drift resulting from changing attacks. Further, we designed a community sharing approach which federates learning through merging models without the need to share sensitive cyber-log data. Finally, by standardizing data and Machine Learning processes in a modular way, we provide network security operators the ability to manage cyber threat events and model sensitivity through community member and analytic method weighting in ways that are best suited for their available resources and data.

Key words. cybersecurity, federated learning, streaming machine learning

AMS subject classifications. 68Q85, 68T05, 68U01

1. Introduction. Computers have been subject to malicious attacks for many years. However, the sophistication and impacts of those attacks have grown dramatically. Successful, timely detection of such attacks is essential to enable defenders to prevent or respond to an attack to mitigate or minimize damage. The consequences from such attacks include substantial financial losses, leaked personal data, and the release of proprietary information or state secrets. The immense human and computational resources required to analyze network logs is prohibitive and often still misses novel attacks. As a result, there is a growing need to incorporate advanced techniques such as machine learning in the cyber defender’s toolbox. However, the cyber domain presents key challenges to traditional batch machine learning techniques [1, 2].

Problem 1.1 The continuous evolution of threat vectors requires continual learning.

Problem 1.2 The volume and sensitivity of network log data prevents long-term retention and sharing across organizations.

Problem 1.3 Barriers-to-entry prevent small or more vulnerable organizations from adopting complex or expensive technology.

Problem 1.4 Feedback from network security operators is important to identify novel attacks and mitigate as quickly as possible.

† Commonwealth Computer Research, Inc., Charlottesville, VA (frank.bentrem@ccri.com, michael.corsello@ccri.com, joshua.palm@ccri.com, https://ccri.com).
In this paper, we present solutions to these problems through an architecture that (1) is streaming, (2) is federated, and (3) manages feedback. The paper is organized in the following way. Our targeted constraints are in section 2, our system architecture is in section 3, the implemented machine learning algorithms are in section 4, and the summary follows in section 5.

2. Targeted Constraints. Individual organizations that train models to recognize cyber threats are severely data limited, however, privacy concerns prevent them from directly sharing raw network data logs. Even anonymized data may collectively reveal sensitive information. We therefore propose a federated learning system [3] where trained standardized models are shared across organizations through their parameters only without sharing any raw data. The models are then merged in a way to improve performance compared to individually trained models. Details on the sharing and merging of trained models are provided in section 4.

In order to entice a large group of organizations to participate in a collaborative learning enterprise, we lower barriers-to-entry by minimizing the computational requirements regarding platform, bandwidth, processing power, and storage. Our design choices were guided by this constraint, where we preferred small algorithms, minimal sharing, and streaming processing. Some modifications to the machine learning algorithms designed for traditional batch processing are needed to enable streaming learning. (For a similar approach see[4].)

3. System Architecture. To achieve the simultaneous goals of supporting real-time analytics of streaming log data and ad hoc offline analytics, we propose a simple directed acyclic graph (DAG) computational model. This model assumes a well-defined data model to facilitate “pluggable” analytics and multi-organizational data federation. The compute flow is based upon a small number of autonomous boxes that are connected in a specific, simple way as illustrated in Figure 1.

![Figure 1. Core computation DAG.](image)

The flow of data through the DAG ensures data at each stage is consistent and provides guarantees of how, when and where data is both shared and integrated. Sharing occurs at “stores” where sharable data is integrated across organizations, as shown in Figure 2, in the live stream using well-known model combiners specific to each analytic shared. In this way, the development of a new analytic involves the creation of a local training processor, a model combining routine, and a prediction processor. Extending the sharing through the use of communities will enable data to be aggregated across community members and then
re-shared as the “community consensus” for further integration at member sites.

4. Machine Learning Algorithms. Finally, any number of prediction algorithms can be plugged into the streaming framework along with methods to merge the separately trained models. We built a prototype for our described system using three example classes of machine learning algorithms, Example 4.1 (neural network), Example 4.2 (naive Bayes), and Example 4.3 (Random Forest), which illustrate how a large number of different classification algorithms might be implemented. The input data for the machine learning algorithms were obtained using existing raw HTTP data logs that were preprocessed to obtain 81-dimensional vectors $x$ (features), most of which were processed according to the method described by Oprea, et al.[5] Limited malicious labels were derived from VirusTotal and FireEye™ where available, and the host traffic ranking was used as a proxy for benign labels. The streaming federated algorithm with scheduled model sharing is described in Algorithm 4.1.

Algorithm 4.1 Streaming federated algorithm

```
while not past time for scheduled sharing do
    Receive HTTP record $R$
    Preprocess to get input vector $x$
    if $R$ contains label information then
        Extract labels $y$
        Perform training iteration with $x \rightarrow y$
    else
        Perform prediction on $x$
    end if
end while
Exchange model parameters with collaborators
Merge models
Repeat from top
```
Example 4.1 (Neural Network) To maintain a small model size, for our neural network model, we avoided deep learning and implemented a multilayer perceptron with five hidden layers of size \([64, 32, 16, 8, 4]\) and output size two for the malicious and benign scores. To merge neural network models we perform a weighted average for the weights and biases of each node in the network. So for each weight \(w_i\) and bias \(b_i\), where \(i\) is the model index, we obtain the merged weight \(w'\) and bias \(b'\) through the weighted averages

\[
(4.1) \quad w' = \langle a, w \rangle \quad \text{and} \quad b' = \langle a, b \rangle,
\]

where \(a\) is the (averaging) weights for the shared models with \(|a|_1 = 1\), and \(w\) and \(b\) are the the weights and biases for all shared models at a given node.

Example 4.2 (Naive Bayes) For the Naive Bayes algorithm, we generate two histograms \(h_{k,i}\) for each feature \(i\), one histogram for the benign labels and one for the malicious labels, where \(k \in \{\text{benign, malicious}\}\). Let \(N_k = |h_{k,i}|\) represent the cumulative numbers of benign and malicious records in the histograms. Then the benign and malicious likelihoods \(L_k\) and evidence \(E\) for each feature \(i\) are given by

\[
(4.2) \quad L_{k,i} = \frac{h_{k,i}}{N_k} \quad \text{and} \quad E_i = \frac{\sum_k h_{k,i}}{\sum_k N_k},
\]

respectively. The probabilities \(p(k \mid x)\) for a log record with feature vector \(x = (x_1, x_2, \ldots, x_n)\) being benign \((k = \text{benign})\) or malicious \((k = \text{malicious})\) are given by

\[
(4.3) \quad p(k \mid x) = \frac{N_k}{\sum_k N_k} \prod_{i=1}^{n} \frac{L_{k,i}(x_i)}{E_i(x_i)},
\]

where \(L_{k,i}(x_i)\) are the likelihoods and \(E_i(x_i)\) is the evidence for the histogram bin corresponding to the value \(x_i\) of feature \(i\). The proof for (4.3) is given in ???. Model merging is accomplished by simply summing the histograms from the shared models.

Example 4.3 (Random Forest) As an example of an ensemble model, the Random Forest algorithm implements some number \(m\) of constituent models that are combined for prediction. The Random Forest algorithm is trained in the normal way. To merge the shared ensembles, we select a sampling of constituent models from each shared ensemble so that we maintain \(m\) models in the merged ensemble.

5. Summary. We presented an automated cyber threat detection system that leverages sharing communities for collaborative, federated learning with a streaming architecture. The computation DAG addresses the need for data privacy and low barriers-to-entry. We completed a prototype with streaming machine learning algorithms with the capability to merge shared models. The results from our internal testing demonstrate the feasibility of our approach and the effectiveness of sharing and merging shared models, however a detailed performance analysis is contingent on more robust labeling.

Acknowledgments. We would like to acknowledge the University of Virginia PCORE-CHASE team for their support.
REFERENCES

[1] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, *On the effectiveness of machine and deep learning for cyber security*, in 2018 10th international conference on cyber Conflict (CyCon), IEEE, 2018, pp. 371–390.

[2] M. Conti, T. Dargahi, and A. Dehghantanha, *Cyber threat intelligence: challenges and opportunities*, in Cyber Threat Intelligence, Springer, 2018, pp. 1–6.

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, *Federated learning: Strategies for improving communication efficiency*, arXiv preprint arXiv:1610.05492, (2016).

[4] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, *River: machine learning for streaming data in python*, 2020, https://arxiv.org/abs/2012.04740.

[5] A. Oprea, Z. Li, R. Norris, and K. Bowers, *Made: Security analytics for enterprise threat detection*, in Proceedings of the 34th Annual Computer Security Applications Conference, 2018, pp. 124–136.
**SUPPLEMENTARY MATERIALS:** Leveraging Sharing Communities to Achieve Federated Learning for Cybersecurity

Frank W. Bentrem†, Michael A. Corsello†, and Joshua J. Palm†

**SM1. Proof of Naive Bayes Probabilities in Equation (4.3).** For a refresher on Bayes’ Rule and independence see [SM2], and a general discussion on the Naive Bayes algorithm can be found in Friedman, et al.[SM1] For simplicity, we assume that each feature in the feature vector \( x = (x_1, x_2, \ldots, x_n) \) is statistically independent of every other feature in \( x \).

Remark SM1.1 While this assumption is often not strictly true, it is generally found to produce well-performing classification algorithms in practice.

Let \( S \) be the set of classes for the classification problem, e.g. \( S = \{\text{benign, malicious}\} \). For each feature \( i \), we generate a histogram \( h_{k,i} \) with identical bins for every label class \( k \in S \). Let \( N_k = ||h_{k,i}|| \) represent the cumulative number of records in each class. Then the class likelihoods \( L_{k,i} \) and evidence \( E_i \) for each feature \( i \) are given by

\[
(LM1.1) \quad L_{k,i} = \frac{h_{k,i}}{N_k}, \quad \text{and} \quad E_i = \frac{\sum_k h_{k,i}}{\sum_k N_k},
\]

respectively.

Claim SM1.1. For a log record with mutually independent features in its feature vector \( x \), the probability \( p(k \mid x) \) of it belonging to class \( k \) is given by

\[
LM1.2 \quad p(k \mid x) = \frac{N_k \prod_{i=1}^{n} L_{k,i}(x_i)}{\sum_k N_k \prod_{i=1}^{n} E_i(x_i)}
\]

in the limit \( N_k \to \infty \), where \( L_{k,i}(x_i) \) are the likelihoods and \( E_i(x_i) \) is the evidence for the histogram bin corresponding to the value \( x_i \) of feature \( i \).

Proof. From Bayes’ Rule with the assumption of independence[SM1], the probability of of a log record with feature vector \( x \) belonging to class \( k \), is

\[
LM1.3 \quad p(k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(k) \prod_{i=1}^{n} p(x_i \mid k),
\]

where \( Z = p(x) = \sum_k p(k) \ p(x \mid k) \). Here \( p(k) \) is the prior probability and \( p(x_i \mid k) \) is the likelihood. By definition, in the limit of a large number of records, \( p(k) = N_k / \sum_k N_k \). Through the same limit, the likelihood is the fraction of records of a class that fall into a given
histogram bin, so that \( p(x_i | k) = \frac{h_{k,i}(x_i)}{N_k} = \mathcal{L}_{k,i}(x_i) \), where \( h_{k,i}(x_i) \) is the histogram bin value for the feature value \( x_i \). Equation (SM1.2) is then equivalent to

\[
(SM1.4) \quad p(k | \mathbf{x}) = Z^{-1} \frac{N_k}{\sum_k N_k} \prod_{i=1}^n \mathcal{L}_k(x_i).
\]

Again under the assumption of mutual independence,

\[
(SM1.5) \quad p(\mathbf{x}) = \prod_{i=1}^n p(x_i), \quad \text{so} \quad Z = \prod_{i=1}^n \frac{\sum_k h_{k,i}(x_i)}{\sum_k N_k} = \prod_{i=1}^n \mathcal{E}(x_i).
\]

Substituting (SM1.5) into (SM1.4) results in

\[
(SM1.6) \quad p(k | \mathbf{x}) = \frac{N_k}{\sum_k N_k} \prod_{i=1}^n \frac{\mathcal{L}_k(x_i)}{\mathcal{E}(x_i)},
\]

which is the result claimed.

\[\blacksquare\]

REFERENCES

[1] J. Friedman, T. Hastie, R. Tibshirani, et al., *The elements of statistical learning*, Springer series in statistics New York, 2001.

[2] L. Metcalf and W. Casey, *Cybersecurity and applied mathematics*, Syngress, 2016.