A Hypergraph-Based Machine Learning Ensemble Network Intrusion Detection System

Zong-Zhi Lin, Thomas D. Pike, Mark M. Bailey, and Nathaniel D. Bastian, Senior Member, IEEE

Abstract—Network intrusion detection systems (NIDSs) to detect malicious attacks continue to meet challenges. NIDS are often developed offline while they face auto-generated port scan infiltration attempts, resulting in a significant time lag from adversarial adaption to NIDS response. To address these challenges, we use hypergraphs (HGs) focused on Internet protocol (IP) addresses and destination ports to capture evolving patterns of port scan attacks. The derived set of HG-based metrics are then used to train an ensemble machine learning (ML)-based NIDS that allows for real-time adaption in monitoring and detecting port scanning activities, other types of attacks, and adversarial intrusions at high accuracy, precision, recall performances. This ML adapting NIDS was developed through the combination of 1) intrusion examples; 2) NIDS update rules; 3) attack threshold choices to trigger NIDS retraining requests; and 4) a production environment with no prior knowledge of the nature of network traffic. 40 scenarios were auto-generated to evaluate the ML ensemble NIDS comprising three tree-based models. The resulting ML ensemble NIDS was extended and evaluated with the CIC-IDS2017 dataset. Results show that under the model settings of an Update-ALL-NIDS rule (specifically retrain and update all the three models upon the same NIDS retraining request) the proposed ML ensemble NIDS evolved intelligently and produced the best results with nearly 100% detection performance throughout the simulation.

Index Terms—Hypergraphs (HG), intelligent systems, machine learning (ML), network intrusion detection, network science.

I. INTRODUCTION

TO DEFEND against unauthorized access an effective cybersecurity system must be supported by machine generated responses to keep pace with a diverse and expanding array of attackers, who are supported by suites of AI enabled tools. The increase in network size and network activities poses considerable challenges for the limited human resources and their comparatively slow pace to adapt against cyberthreats. Furthermore, the rapid advances in Internet and communication technologies continue to introduce increasingly complex dynamics comprising of more variants of cyberthreats, such as malware, phishing, port scanning, and denial-of-service attack, to name just a few. These rapidly evolving network security threats and the need to detect, characterize and defend against diverse and adapting cyberattacks necessitate the integration of machine learning (ML) techniques into cybersecurity practices [1].

ML techniques can be categorized into the two seminal categories as either supervised or unsupervised learning algorithms [2]. Supervised learning algorithms learn from labeled data (e.g., data labeled as known intrusion attempt), whereas unsupervised learning algorithms statistically assess unlabeled data, based on the features of that data. These ML methods often require large amounts of network traffic data with laboriously developed data features to train network intrusion detection systems (NIDSs) [3], [4]. To overcome this training data challenge, transfer learning can be employed to leverage the curated data from network traffic and apply it to another, without necessarily needing to use the same ML approach [5].

ML-based NIDS continue to struggle to develop approaches that effectively integrate ML into NIDS [6], [7]. A grave challenge to NIDS has been its vulnerabilities to ML generated attacks that escape the detection with a high level of success [8], [9]. Furthermore, NIDS involving ML are often trained and developed offline causing a time lag in fixing vulnerabilities as attacks spread across the network [10]. ML-based approaches have proven effective at developing increasingly successful attack vectors, but struggle to develop effective defense approaches. The increasingly heterogeneous and complex dynamics of the cyber-attacks, many produced through ML approaches, requires cyber-defenses to effectively integrate ML into their strategies.

A fundamental challenge for developing an effective ML-based NIDS is overfitting. In order for a deployed ML-based NIDS model to generalize to novel attacks, it must be optimally tuned to prevent it from performing too well on its training data [11]. The reality in ML is that one should never trust a ML model that gets 100% of its training data correct because that model will perform very poorly when faced with novel attacks, a phenomenon known as the bias-variance tradeoff. This poses a significant risk to network security, and presents a daunting challenge in devising effective ML-based NIDS to defend against ML-based attacks.

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. 54, NO. 11, NOVEMBER 2024 6911

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Our work seeks to aid NIDS development through the integration of a hypergraph (HG) metrics and a ML ensemble approach focused on port scan attacks. Integrated into this approach is auto generated novel attacks the NIDS must continue to identify and defend against. First, we use a variety s-closeness centrality metrics to capture the adversary’s signature pattern of clustered malicious activities, primarily port scans, on a targeted machine. Second, we investigate multiple ways to render these HG metrics as features augmenting the NIDS training dataset. Finally, we devise a methodology to create an intelligent cybersystem to continuously train and update a ML ensemble NIDS in real-time within the modeled network.

II. RELATED WORKS

ML models are applied across a variety of domains to include cybersecurity, with varying effectiveness. In the context of digital world, the attacker launches an attack on victims through the computer network; therefore, it is natural to research cybersecurity from the perspective of a network problem [12]. One can model the network traffic data from source and destination Internet protocols (IPs), source and destination ports as nodes in a network, while the protocols and attacks can be modeled as links. Network science-based approaches have proven effective in understanding the system traffic to find attack information, detect abnormal traffic, or devise network detection and prevention approaches [13].

Although network science is ubiquitous in cybersecurity, limited work explores the utility of HGs, which is an extension of network science to algebraic topology, to aid cybersecurity efforts and in this case, specifically the intrusion detection problem. Earlier studies model the multistep of logged cyber activities (e.g., remote access, directory scanning, SQL injection and Web Server Control to attain Denial of Service goal) as a HG to enable the detection of subsequences that represent an intrusion [14], [15]. Guzzo et al. [15] showed that the intrusion detection problem defined over their cyberattack model is a NP-Complete problem.

A few network intrusion studies have used HGs to improve the performance of ML-based NIDS to detect several specific types of intrusions, such as port scan, Brute-force, heartbleed, botnet, Denial of Service, Web attacks, and Distributed Denial of Service (DDoS). The study in [16] uses HGs to identify the key features for NIDS by way of modeling the multiway relationships among minimal sets of explanatory attributes resulting from dimensionality reduction with respect to the degree of dependency among the explanatory attributes. The study in [17] exploits properties of HGs to optimize its genetic algorithm performance for parameter setting and feature selection in a support vector machine model to balance evaluation criteria of intrusion detection rate, false alarm rate and number of features. The HG-based study in [12] uses a HG as an intermediate construct to derive a number of latent features, but not to reveal concerning patterns of network activities. The study in [18] proposes a HG clustering model based on the a priori algorithm to describe the association between fog nodes which are suffering from the threat of DDoS, thereby increasing resource utilization rate of the system. Joslyn et al. [19] used a HG approach using Domain Name Systems cyber data to represent the joint relationships between collections of domains (coded as hyperedges) and IP addresses (coded as vertices), enabling both analytic and visual explorations of its s-components and distributions of these component sizes in terms of node and edge counts to find abnormal IPs and domains and examine the neighborhood of known bad IPs or domains. However, Joslyn et al. [19] did not use HG-based metrics as part of the features to train ML NIDS yet.

Another facet of NIDS challenges is the need for automated updates and deployment to rapidly mitigate vulnerabilities and stop the spread of infiltrated attacks into other parts of the network. ML-based approaches are uniquely suited to support automatic updates. The study in [20] implements a Naïve Bayesian classifier to balance detection for different types of real-time networking attacks, such as majority intrusion of Denial of Service and minority intrusion of user to root (U2R) in real network connections datasets. The study in [21] used a decision tree and selected 12 essential features of network to implement a real-time intrusion detection system to distinguish normal network activities from main attack types (Probe and Denial of Service) with a detection rate higher than 98% within 2 seconds. The study in [22] implements a real-time NIDS based on a deep neural network to identify intrusions by analyzing network data in real-time, achieving at 81% for accuracy in testing performance results. The challenge is that the real-time ML NIDS in these studies and many others continue to miss some intrusions at some small likelihood.

As ML becomes integrated into cyberattacks an additional aspect any NIDS must consider is the use of adversarial training to automatically adapt intrusion attempts to evade detection and be classified as legitimate or normal [23], [24]. Optimization models have been rigorously applied in this data augmentation approach but have shown limited success in enhancing cyberdefense [25]. In addition, experiments have also shown evolutionary approaches in constrained environments to generate adversarial examples to evade NIDS, mimicking the learning behavior of potential attackers to effectively cause high-misclassification rates for more than ten commonly used ML models [8].

Previous research has used HGs and ML in various forms and with varying degrees of sophistication to enhance cybersecurity. From a more general perspective, these research efforts have a common thread that treats robustness (the ability to withstand an attack) and resilience (the ability to recover after being successfully attacked) as mutually exclusive goals, while generally favoring robustness. Our research combines HG metrics and ML automation, in an effort to develop and evolve a NIDS that can both withstand attacks, but also recover quickly when penetrated.

III. EVALUATION METHODOLOGY

To devise and evaluate the novel ML ensemble NIDS, our evaluation methodology comprises several connected components, including 1) NIDS evaluation framework consisting of updating NIDS detection performance with HG and updating NIDS adversarial attacks; 2) NIDS model performance
evaluation; and 3) research dataset. These components allow for the iterative simulation of the evolutionary fight of the deployed NIDS models and the adapting cyber-attacks using generative adversarial examples.

A. NIDS Evaluation Framework

A set of virtual computer agents connected with simple rules of interaction simulates the computer network used for this research, as illustrated with the network of three computers in the green circle in Fig. 1. The ensemble NIDS model within the network takes and renders the features of network traffic data to determine if it is an intrusion. To simplify the interactions among computers, the computers deploy with the same set of NIDS, accordingly each can be updated with the same retrained NIDS or the set of retrained NIDS.

The Adversarial Example Generation module applies state-of-the-art techniques to generate and introduce adversarial examples to simulate the unseen adversarial attacks to the network of computers. This enables data augmentation for the NIDS model training, thereby auditing or enhancing the adversarial robustness of deployed NIDS models. Successful intrusions are sent to the Network Traffic Database module to serve as part of train dataset to retrain NIDS models.

The Network Traffic Database module collects network traffic data, NIDS classified “Attack” samples, and failed-to-detect adversarial examples. Whenever a request to retrain any NIDS model is triggered, the Network Traffic Database module provides a new training dataset mixing normal network traffic data adequately proportionate to the size of cumulative evaded successful intrusions at the time of NIDS model retraining request.

The Behavioral Analytics module applies HG analysis to provide systemic oversight over a longer time span at each computer or the entire network to detect intrusions sometime after the malicious, including port scan and infiltration.

The NIDS Model module retrains and updates NIDS to improve detection performance level, whereas the Computer Module maintains an ensemble of ML-based NIDS to differentiate normal network users from cyber-attacks. Note that infiltrated cyber-attacks in a computer can spread to other computers within the simulated network. The Network of Computers module coordinates the NIDS updates from the NIDS retraining request at other computers within the network [26] and coordinates with the Behavioral Analytics module to keep the ML ensemble NIDS updated.

The Scorecard module collects and compute the performance of deployed NIDS in terms of the total of infiltrated malicious attacks and failed-to-detect adversarial examples and its associated confusion matrix at each computer over time. When the total of infiltrated malicious attacks exceed a specified threshold choice, a request to retrain and update a NIDS model or a NIDS ensemble model will be triggered to enable evidence-based decisions and maintain and/or improve performance of deployed NIDS models in the intrusion detection accuracy. The deployed ML ensemble NIDS models continue to defend the system if there is no retraining request. The evaluation process continues and stops when it reaches the specified simulation time.

B. NIDS Model Performance Evaluation

ML models are typically developed in two stages; a training stage to develop the model and a testing stage to evaluate the developed model. Likewise, the research data is split using the 80% and 20% convention between training and testing stages. Individual outputs of ML models are generally classified using numerical likelihood. If the classification is a binary choice of A and B, then when the likelihood is very close to 1.0 for category A, the interpretation is that the model predicts Category A to realize with very high confidence. If the likelihood is very close to 0.0, the model predicts Category A to happen with very low confidence. If the likelihood falls around 0.5, implying high uncertainty, this often requires further review of the data to support the eventual prediction. Built on raw likelihood scores, we adopt the commonly used confusion matrix and mean F1-score to evaluate the performance of the trained ML models as well as the proposed ML ensemble NIDS.

C. Research Dataset

We use the intrusion detection evaluation dataset, publicly released in 2017 from the Canadian Institute for Cybersecurity (CIC-IDS2017). The CIC-IDS2017 data set consists of several attack classes, to include: port scan, Brute-force, Heartbleed, Botnet, Denial of Service, DDoS, Web attacks, and infiltration of the network from inside [27], [28].

Our work considers the port scan attack class first and extends the results to detect other attack classes. Out of its total 84 features in the CIC-IDS2017 port scan dataset, eight of them are understood to be most important features on detecting attacks: 1) duration of the flow in microsecond; 2) total packets in the forward direction; 3) total packets in the backward direction; 4) total size of packet in forward direction; 5) total size of packet in backward direction; 6) number of flow bytes per second; 7) number of flow packets per second; and 8) download and upload ratio [12]. Out of its total 286,487 labeled records, total 407 records with negative duration of the flow in microsecond, or missing or infinite values for at least one of those eight features were removed from the research dataset. In addition, all the malicious attacks were relabeled as “PORT SCAN” to simplify the coding complexity while accomplishing the same level of detection performance. The ratio of labeled port scan attacks and benign records in the resulting research dataset of 286,060 labeled records is about 55.5% and 44.5%, respectively.

The feature of transport protocol is included as a categorical variable because the dataset indicates the number of port scan events via the transport protocol six (namely, transmission control protocol, TCP) is 158,797, which is about 2.37 times of the 67,053 benign events via the same transport protocol. Two other values of the protocol feature are 0 (version 6 of the IP, IPv6) and 17 (user datagram protocol, UDP). We use the set of nine network raw features (NRF, the protocol feature and the eight features listed above) as the base training dataset to train the base NIDS model in the ML ensemble NIDS. As depicted by Table I summarizing basic descriptive statistics for each numeric feature of this base training dataset NRF by record type of Port Scan and BENIGN, there exists noticeable
differences in the statistics among those eight features to provide their differentiating power to detect network activity type. This set NRF will be expanded further with HG-based features to be derived later based on another three network features of source IP, destination IP, and destination port number. Source port number is typically randomly assigned, thus it is not included for further uses.

Three additional practices were applied to expand, diversify and toughen the malicious attack dataset to improve the development of the proposed ML ensemble NIDS: 1) leverage the CIC-IDS2017 dataset to generate a dataset simulating the case where multiple hacker source IPs port scans multiple destination IPs; 2) apply an adversarial example generation technique on the CIC-IDS2017 dataset to enable data augmentation needed to conduct adversarial training; and 3) relabel all other malicious attacks as “PORT SCAN.”

IV. DEVELOPING HYPERGRAPH-BASED ML ENSEMBLE NIDS

Network traffic flow data typically includes source and destination IPs, source and destination ports, and the communication protocol, as found in Network Intrusion Detection Evaluation Dataset CIC-IDS2017 [27] used for this study. This data was then abstracted into a HG to capture the pattern of port scan activities. Mirroring the concept of using closeness centrality to measure the proximity of one node to all others in a network, the abstracted HG enables to formulate a set of the s-closeness centrality to quantify the pattern. These results create a set of HG metrics to use in training the NIDS models, thereby becoming the building blocks of the proposed ML ensemble NIDS. We then extend these results to develop a behavioral analytic capability monitoring the network activities with no prior knowledge of their activity types to assist with online detection of malicious, including port scan and activities. It is worth noting that modeling the multidimensional cyber-security data as HGS, the visualization of HGS and the computation of HG metric features were accomplished using the HyperNetX library [29], [30].

A. Model Port-Scanning Activities With Hypergraph

A basic understanding of HGS is necessary to appreciate the metrics of interest. A HG comprises a set of vertices \( V \) and set of hyperedges \( E = \{e_1, e_2, \ldots, e_m\} \) such that each hyperedge is a subset of vertices set \( V \), namely, \( e_i \subseteq V \). A typical graph or network is a HG where all its hyperedges have size of two vertices, i.e., \(|e_i| = 2\). An \( s \)-walk of length \( k \) between hyperedge \( f \) and \( g \) is a sequence of hyperedges \( f = e_{i_0}, e_{i_1}, e_{i_2}, \ldots, e_{i_k} = g \) such that for \( j = 1, \ldots, k \), we have \(|e_{i_j-1} \cap e_{i_j}| \geq s \) and \( i_{j-1} \neq i_j \) [31]. An \( s \)-path is an \( s \)-walk where hyperedges are not repeated. The \( s \)-distance between hyperedge \( e \) and \( f \) is the length of its shortest \( s \)-path, denoted as \( d_s(e, f) \). In addition to length, each \( s \)-path in a HG has width, the minimal number of common vertices among all the consecutive hyperedges of the \( s \)-path. In other words, the width of an \( s \)-path is at least \( s \) since all consecutive intersections are at least size \( s \). An \( s \)-component in a HG is a collection of hyperedges so that any pair of hyperedges are connected via an \( s \)-path.

In the context of port scanning activities, let \( S_{s-ip} \), \( S_{d-ip} \), and \( S_{d-port} \) be the sets of all source IP addresses, all destination IP addresses and all destination ports, respectively. Let \( E \) be a set of hyperedges in which each hyperedge is either source or destination IP address that is associated with port scanning a set of destination ports in a stream of network activities. The

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE I
SELECTED FEATURES DESCRIPTIONS FROM CIC-IDS2017 PORT SCAN DATA FILE TO DEFINE THE RESEARCH DATA SET

| Feature Description                                      | Data Type | Mode | Maximum      | Average     | Std. Dev.     |
|----------------------------------------------------------|-----------|------|--------------|-------------|--------------|
| Duration of the flow in microsecond                      | Port Scan | 47   | 119809735    | 82885.94    | 2326775.89   |
|                                                          | Benign    | 30985| 119999949    | 5386984.20  | 31562986.85  |
| Total packets in the forward direction                   | Port Scan | 1    | 150          | 1.02        | 0.43         |
|                                                          | Benign    | 2    | 3119         | 6.55        | 28.98        |
| Total packets in the backward direction                  | Port Scan | 1    | 30           | 1.00        | 0.15         |
|                                                          | Benign    | 2    | 3635         | 6.67        | 42.23        |
| Total size of packet in forward direction                | Port Scan | 0    | 1473         | 1.09        | 5.66         |
|                                                          | Benign    | 68   | 232349       | 524.05      | 2771.69      |
| Total size of packet in backward direction               | Port Scan | 6    | 11595        | 12.24       | 267.40       |
|                                                          | Benign    | 142  | 7150819      | 6079.02     | 76351.31     |
| Number of flow bytes per second                          | Port Scan | 139535| 80000000     | 220359.75   | 459863.69    |
|                                                          | Benign    | 5684 | 2070000000   | 2241033.31  | 38472283.25  |
| Number of flow packets per second                        | Port Scan | 42553| 20000000     | 62690.57    | 127930.00    |
|                                                          | Benign    | 113  | 30000000     | 62501.33    | 247879.03    |
| Download and upload ratio                                | Port Scan | 1    | 2            | 0.99        | 0.09         |
|                                                          | Benign    | 1    | 124          | 0.67        | 0.62         |

Fig. 2. This HG with 15 edges and 34 vertices was constructed by 43 records with benign and port scan attack class label denoted as B and ∧, respectively. The ports 21, 53, 443, 56344, 8613, 35066, 42154, and 55107 in normal uses by source and destination IPs were also part of smaller set of centers to concentric circles associated with source or destination IPs.

The constructed HG can be denoted as $HG = \{V = S_{d-port}, E = S_{s-ip} \cup S_{d-ip}\}$, so each unique IP or port is only included once in the HG. Thus, this constructed HG summarizes the structure of interactions and connectivity among IPs and destination ports involved in network activities modeled.

Each hyperedge in $E = S_{s-ip} \cup S_{d-ip}$ is mathematically defined as a subset of $S_{d-port}$ based on its associated destination port scanning activities in a network. For example, the edge of source IP 173.194.208.155 at the top right corner of Fig. 2 comprises a single node of destination port 35066 reflecting its access in a normal network activity. As depicted in Fig. 2 with a small HG example, the pair of Source IP-172.16.10.1 and Destination IP-192.16.10.50 addresses were associated with a long and common list of available destination ports, the crowded black dots in Fig. 2, involved in the network activities modeled [28]. It happens that the attacker from source IP 172.16.0.1 port scanned the targeted destination IP 192.168.10.50 via a list of destination ports, including ports 53 and 80, that were used normally by other source IPs.

The $s$-closeness centrality $C_s(e)$ for an edge $e$ measures how close $e$ is to all others in its $s$-component. It is computed by the following formula $C_s(e) = (|E_s| - 1)/(|\sum_{f \in E_s} d_s(e,f)|)$, where $E_s$ is the set of hyperedges in its $s$-component that contains the edge $e$ [31]. It can be readily verified that the $s$-closeness centrality for each hyperedge in the $s$-component comprising only concentric circles is one. As shown in Fig. 3, for $s \in \{11, 12, \ldots, 30\}$, $C_s(e) \in \{172.16.0.1, 192.168.10.50\} \rightarrow 1$, and the two edges of IP...
addresses almost form two concentric circles based on the set of commonly accessed destination ports. Coincidentally, the source IP is the hacker used to scan the destination ports available to the destination IP in the training dataset. Therefore, these individual s-closeness centrality metrics essentially capture the hacker’s signature pattern of port scan activities on a targeted machine and have potential to render features to expand the NIDS training dataset.

We remark the s-closeness centralities for the benign network activities vary between zero and one before the value of s hits 22. After it hits 22, the majority of them stay at zero and some at one, thereby yielding a decreasing trend of its mean s-closeness centralities to close to zero, as indicated by the blue curve in Fig. 4. To expand the proposed ML ensemble NIDS to detect both benign and other types of attacks with high performance, this decreasing trend of its mean s-closeness centralities proves in our numerical evaluations to be effective in encoding the pairs of nonhacker IPs with a set of weights approximating the distribution of their average s-closeness centralities.

B. Generating Hypergraph-Based Feature Sets

We now formulate the mathematical strategy to compute s-closeness centralities of the edges in the HG with a properly specified skip-interval k. The strategy creates two different sets of HG-based features to use in training the NIDS models. Both feature sets augment the dataset of the aforementioned nine raw network flow features (NFFs).

Let \( C_{3+n	imes k}(e) \) be the \((3 + n \times k)\)-closeness centrality of the hyperedge e for a positive integer n. It often takes much longer time to compute \( C_1(e) \) and \( C_2(e) \) (especially for a much larger HG), yet when \( s \in \{1, 2\} \), the first two s-closeness centralities are more likely to contain signals from the normal uses of the destination ports by other hyperedges, thereby statistically causing the first two s-closeness centralities to reduce its differentiating power if there exists a pattern of port scan activities. In the numerical evaluation, let the values of

\[ n \in S = \{0, 1, 2, \ldots, 9, 10\} \]

and the value of skip-interval k is subject to the condition that the value of \((3 + 10 \times k)\) is about 70% of the maximal total number of nodes or destination ports connected to hyperedges (namely, source IP or destination IP).

It is by design to evenly spread out the computation of these s-closeness centralities because the values of latter s-closeness centralities (namely, at larger s) carry more signal if there exists a pattern of port scan activities. For computational design convenience and completeness, let all the remaining s-closeness centralities be zero (0) at hyperedge e when the value of s is greater than its total number of nodes or destination ports but is smaller than the value of \((3 + 10 \times k)\).

Adopting the above mathematical definitions and symbols, we construct two different HG feature sets for training NIDS and their numerical evaluations.

1) Feature set HGI includes the set of nine raw features (RFs) and all the individually \((l)\) computed s-closeness-centralities \( C_{3+n	imes k}(e) \; (n \in S) \) and their sum \( (\sum_{n \in S} C_{3+n	imes k}(e)) \) for each hyperedge e. The resulting dataset HGI is used to train the NIDS model HGI.

2) Feature set HGA includes the set of nine RFs, the last computed the \( n \)-th s-closeness-centrality (namely, \( C_{3+10	imes k}(e) \)) and four aggregate (A) HG measures. The resulting dataset HGA is used to train the NIDS model HGA. The four aggregate HG measures are a) the sum of all the computed s-closeness-centralities (i.e., \( \sum_{n \in S} C_{3+n	imes k}(e) \)); b) the total number of unique destination ports associated with source IPs hyperedge (i.e., \(|e_{s-ip}|\)); c) the number of destination ports associated with destination IPs hyperedge (i.e., \(|e_{d-ip}|\)); and d) the sum of destination ports tied to source or destination IPs (i.e., \(|e_{s-ip}| + |e_{d-ip}|\)).

C. Hypergraph-Based ML Ensemble NIDS Model

The proposed HG-based ML ensemble NIDS model comprises three classifiers based on Random Forest and light
Algorithm 1 Process to Detect Potential Port Scan Activities

Step 1: Construct the hypergraph with network traffic data of their source and destination IPs and destination ports.

Step 2: Compute its eleven (11) evenly spaced $s$-closeness centralities for each edge $e$, namely, $C_{s+clos}(e), n \in S$.

Step 3: Make each of last six $s$-closeness centralities as zero if it is less than 0.95 or 1 otherwise.

Step 4: Flag port scan activities if the sum of last six converted $s$-closeness centralities is greater than or equal to two and the pair of source and destination IPs has not been flagged yet.

It is shown numerically that the $s$-closeness centralities for $s \in \{1, 2\}$ can take a very long time to compute in a large HG. Thus, the size of network traffic data used to construct a HG to implement this behavioral analytics capability needs to evaluate and set properly in order to receive timely alerts and/or initiate NIDS retraining requests to update the ML ensemble NIDS. In addition, the choices of both parameters in steps 3 and 4 were based on limited evaluations numerically, requiring thorough investigations before putting them into any operational uses.

V. Evaluating Hypergraph-Based ML Ensemble NIDS

Conducting the evaluation of the proposed ML ensemble NIDS according to the simulation process depicted in Fig. 1, the Evaluation Framework for the ML ensemble NIDS, this section comprises evaluating the effectiveness of adversarial examples to cause target NIDS to misclassify, NIDS models, and the ML ensemble NIDS, respectively. The ML ensemble NIDS comprises three pretrained and active NIDS (namely, RF, HGI, and HGA) and retraining another NIDS when the total cumulative number of infiltrated attacks at each computer or entire network during the simulation exceeds the threshold choice, such as two attacks.

In general, the retrained NIDS will replace the worst of the active NIDS only if its detection performance based on certain metrics, such as accuracy, is better than the worst of the three active NIDSs, or referring it as Forgo-the-Worst NIDS. Another option is to retrain and replace all the three active NIDSs correspondingly and, respectively, or referring it as Update-ALL-NIDS by position and model.

Numerical evaluations are performed in two different settings. The first and reduced setting evaluates the proposed ML ensemble NIDS framework with attacks against the ML ensemble NIDS from the port scan only network traffic data along with the set of adversarial examples generated. Normal and benign network traffic activities in the port scan dataset are not used for attacking but will be used to produce new training dataset with a balanced mixing of normal network traffic data and cumulative evaded successful intrusions at the time of NIDS model retraining request. The second and complete setting evaluates the ML ensemble NIDS with all the attacks of the CIC-IDS2017 dataset and the set of adversarial examples generated.

A. Adversarial Example Generation and Effectiveness

To evaluate the NIDS model robustness, research on robustness has recently advanced to assessing how susceptible models are to adversarial manipulation [9]. To account for adversary adaptation our model uses the Zeroth-order-optimization (ZOO) attack method against it to generate new attack examples [33]. Studies show that training ML models on adversarial examples provides better performance [34]. Thus, this work uses adversarial generated data to address the feature vulnerabilities and make incremental improvements on the deployed NIDS models. This helped us examine and enhance the ML ensemble NIDS adversarial robustness [8]. To simplify...
the development and testing of the ML ensemble NIDS, we use only the reduced dataset of the port scan network traffic in the evaluation.

Our model approach does assume the adversary does not know that the ML ensemble NIDS uses HG \( s \)-closeness centralities metrics, and so the adapting NIDS is trained with the nine RFs using a light gradient boosting machine [32]. The ZOO attack, which estimates the gradients of the target model using zeroth order stochastic coordinate descent along with dimension reduction, hierarchical attack and importance sampling techniques, perturbs these RFs to attack the substitute NIDS and generate adversarial examples [33]. Specifically, the process in Algorithm 2 outlines the steps to generate adversarial examples from a set of about \( 114900 \) Port Scan activities sampled from the CIC-IDS2017 Port Scan dataset, assuming Hacker does not know the model uses HG \( s \)-closeness centralities metrics. Out of total 17188 adversarial examples generated via Algorithm 2, 14799 of them with predictive Port Scan probabilities greater than or equal to 0.55 were used in the numerical evaluations.

To assess the effectiveness of these adversarial examples, these 14799 examples were fed into four different NIDS models (substitute NIDS and the deployed NIDS models of RF, HGI, and HGA) to compute and construct their respective distributions of port scan detection scores. Their respective results yield detection performances in the lowest performance to highest performance of Models A, B, C, and D, as depicted and summarized in Fig. 5. The ranking of the four models is based on the adoption of port scan threshold score 0.5 (the dotted yellow horizontal line) to adjudicate if a network activity is a port scan.

These curves of detection scores also suggest that there might have some combinations of those individually computed \( s \)-closeness centralities and their aggregates to include in the NIDS model training data set to help defend adversarial examples, given that the aggregate metrics of those \( s \)-closeness centralities metrics have captured the pair of IPs launching the port scanning attacks.

### B. Effectiveness of Hypergraph-Based NIDS Models

To train a NIDS model for port scanning activities, as stated in an early section, nine out of 80 features in the CIC-IDS2017 port scan data file has been evaluated and commonly selected to be most important features on detecting attacks [12]. Therefore, we use the feature set HGI comprising a set of HG metrics and those nine features in the 40% sample of the CIC-IDS2017 port scan dataset to train and build a tree-based NIDS. As indicated in Table II, the derived set of HG \( s \)-closeness-centrality metrics HGI enhances our NIDS model to reach nearly 100% precision, recall and \( F_1 \) score performances.

### C. Evaluating the ML Ensemble NIDS With Port Scan Data

This research then evaluates the ML ensemble NIDS in six cases with generic settings over a simulated network of 10 computers over 30 epochs. The threshold choice of infiltrated port scan attacks to start an NIDS retraining will be set and evaluated at each value in the set \( TH = \{2, 5, 10, 20, 30, 40, 50, 100\} \). Table III specifies the six scenarios in the following factors.

1) **IP PAIRS**: Pairs of source and destination IP.
2) **NIDS**: Rule to update NIDS—not allowed or allowed to update and forgo the worst NIDS (FTW) or to update all the NIDS models (UALL).
3) **ADV. Ex.**: Include adversarial examples or not.
TABLE III

| CASE | IP PAIRS | NIDS  | ADV. EX. | THRESH. | PROD. |
|------|----------|-------|----------|---------|-------|
| 1    | 1        | Static| No       | TH      | No    |
| 2    | 16       | Static| No       | TH      | No    |
| 3    | 16       | FTW   | Yes      | TH      | No    |
| 4    | 16       | UALL  | Yes      | TH      | No    |
| 5    | 16       | UALL  | Yes      | TH      | No    |
| 6    | 16       | UALL  | Yes      | TH      | Yes   |

4) **THRESH.**: Values of threshold in the set \( TH \) to trigger NIDS retraining request(s).

5) **PROD.**: Emulate production environment with no prior knowledge of the nature of network traffic or not.

Case 1 assumes knowing *a priori* that port scanning activities are from a known pair of source and destination IPs. The ML ensemble NIDS stays static throughout the simulation. Numerical results indicate that the ML ensemble NIDS yields the 0% detection performance of false negative percentage (FNP) throughout the evaluation window, namely, misclassifying zero attack as normal traffic. The NIDS HGI and HGA in the ML ensemble NIDS help detect a small set of port scan activities that the NIDS RF fails to detect, thereby enabling the ML ensemble NIDS to attain perfect detection score and zero FNP throughout the simulation.

Case 2 extends case 1 to include port scan activities from 15 other pairs of pseudo and unknown source and destination IPs. The ML ensemble NIDS continues to stay static throughout the simulation.

The near 100% detection performance of the ML ensemble NIDS model in the FNP in Fig. 6 indicates that the NIDS HGI and HGA help detect most port scan activities from multiple pairs of pseudo and unknown source and destination IPs that the NIDS RF fails to detect as indicated by the Curve C in Fig. 5.

It is seemingly the level of adversarial robustness in the ML ensemble NIDS model is enhanced because the negative impact of simulated multiple pairs of unknown source and destination IPs on its detection performance is not as severe as originally expected from the observations in Fig. 6. This might be partly because the values of the nine NFFs remain intact, and the tree-based NIDS based on these NFFs has a decent detection performance. A more realistic dataset of port scan attacks from unknown source to destination IPs is needed to study the impact of varying pairs of IPs on the detection performance of the ML ensemble NIDS model.

Case 3 extends case 2 to allow the retrained NIDS HGI (when infiltrated events exceeds certain threshold choice) to replace at most one of the three active NIDS in the ML ensemble NIDS with the FTW rule, namely, when its detection rate of port scan events, is higher than the worst of the three NIDS models. Its detection performance becomes somewhat volatile in the beginning and near the 150th run but stabilized after about 150th run, as depicted in Fig. 7.

Case 4 extends case 3 to include attacks from adversarial examples generated with the ZOO attack method. Its detection performance greatly deteriorates in the beginning, as depicted in Fig. 8. The deterioration causes the number of infiltrated attacks to exceed the study threshold choice in \( TH \), thereby triggering the first NIDS HGI retraining request to replace the active NIDS with the lowest-detection rate according to the FTW rule. After the first NIDS retraining and replacement, its detection performance gradually becomes somewhat stable and maintains a higher-detection rate. In addition, the set of adversarial examples continues to infiltrate the ML ensemble NIDS, thereby triggering more NIDS HGI retraining requests to replace and update the active NIDS in the ML ensemble NIDS. Therefore, detection performance of the ML ensemble NIDS via Forgo-the-Worst-NIDS update rule for case 4 is sensitive to the size of attack threshold choice, causing vulnerability and plausibility in deployment.
Case 5 extends case 4 but changes its NIDS update rule. Upon triggering an NIDS retraining request, case 5 allows retraining and replacing all the active NIDS correspondingly, namely, UALL, thereby retraining and updating the NIDS model RF, HGI, and HGA, respectively, at the same run and time. As expected, its detection performance greatly deteriorates at the first computer and first epoch, delivering FNP at 29%, as depicted in Fig. 9. However, after completing the first NIDS retraining request to update and replace the whole ML ensemble NIDS, the deterioration stops on the second computer at the first simulation epoch, and then it attains perfect detection score throughout the rest of simulation. Therefore, the sustained perfect detection performances of the ML ensemble NIDS for case 5 suggests that the Update-ALL-NIDS update rule mitigates the ML ensemble NIDS model’s sensitivity to the size of attack threshold choice.

Case 6 extends case 5 to simulate production environment with no prior knowledge of the nature of network traffic. It uses HG construct to detect if there exists any signature pattern of port scanning activities. Given no prior knowledge of the nature of network traffic, selecting the retrained NIDS with higher-detection rate to update the active NIDS is in alignment of the commonly adopted zero-trust practice in cybersecurity. Its detection performance is the same as that of case 5, namely, volatile initially but attaining 100% detection level after updating the ML ensemble NIDS at all the threshold choice in TH.

D. Evaluating the Ensemble NIDS With CIC-IDS2017

This section extends to evaluate the ML ensemble NIDS with the second setting, namely, the entire CIC-IDS2017 dataset and the set of adversarial examples generated, with case 5 scenario that allows retraining and replacing all the active NIDS correspondingly. In addition to the port scan (260 293) and BENIGN (2 430 674) activities, there are 13 other types of attacks, comprising DoS Hulk (235 435), DoS GoldenEye (10 540), DoS slowloris (5 920), DoS SLOWHTTPTEST (5 635), DDoS (131 016), FTP-Patator (8 110), SSH-Patator (6 031), Bot (2014), Web—Brute force (15 38), Web attack—XSS (667), Web attack—Sql injection (22), infiltration (36), and Heartbleed (11).

In this extension, the NIDS Model Module is accordingly modified to retain existing NIDS models if any of their detection performances in F1 Score is higher than the best of the newly retrained NIDS models. In addition, the authors encoded the pairs of nonhacker IPs in each record of the CIC-IDS2017 dataset with a set of weights approximating the distribution of average s-closeness centralities (as depicted by the blue curve in Fig. 4) to help differentiate benign activities from other types of attacks. The set of weights approximating its 11 features of average s-closeness centralities for any nonhacker pairs of IPs is

\[
(0.2, 0.15, 0.1, 0.05, 0.04, 0.03, 0.02, 0.01, 0.008, 0.006, 0.005).
\]

The total 2,668,152 benign and malicious attack traffic (comprising of the 667 038 attacks and 2,001,114 BENIGN activities) were randomly sampled and split into 300 batches, roughly 8 900 records for each batch (comprising 25% attacks and 75% BENIGN activities) to feed into the simulation over a network of 10 users in 30 epochs. FNP, accuracy, precision, recall and F1 score were computed and evaluated throughout the simulation.

The numerical results summarized in Fig. 10 shows that the detection performance of the proposed ML ensemble NIDS model based on F1 score is not stable when the thresholds to trigger model retraining were set at 10 attacks or lower. This suggests more frequent model retraining does not necessarily result in effective model learning. However, when the thresholds to trigger model retraining were set at 20 attacks or higher, the results depicted in Fig. 11 show that its detection performances are highly effective in F1 Scores.

To illustrate the proposed ML ensemble NIDS model’s robustness against and resiliency to the attacks in the CIC-IDS2017 dataset, the baseline model results were obtained without using the HG-based features in case 5 simulation setting that starts with a set of three identical NIDS models trained solely on NRF dataset and then follows with retraining NIDS NRF models separately. The baseline model results, as depicted in Fig. 12, improve initially but deteriorate eventually in most threshold choices in the simulation. Therefore, with proper choices of threshold, our numerical results provide evidence that the features of the s-closeness centralities of dynamic HGs help the tree-based ML models, initially trained only on port scan activities, to learn and improve over time to give near perfect detection performances, furnishing both robustness and resiliency in the proposed ML ensemble NIDS model.
E. Numerical Results Summary

Cyber-attacks are occurring across more heterogeneous and complex vectors. Numerical results indicate that the ML ensemble NIDS had the best and near perfect performance over port scan dataset for cases 5 and 6—comprising an ensemble of three NIDS along with Update-ALL-NIDS rule when needed. The significant improvement of the ML ensemble NIDS indicates that the two sets of HG-based features, namely, HGI and HGA, derived from the higher-level interactions that the HG models are indeed meaningful. These settings also allow the ML ensemble NIDS to be insensitive to the size of attack threshold choices for the reduced setting with port scan data only, as partially depicted in Fig. 13. The ML ensemble NIDS’ insensitivity to the threshold size retains at slightly higher values.

The consistent and near perfect $F_1$ Scores obtained from evaluating the ML ensemble NIDS with the CIC-IDS2017 dataset illustrate that the uses of the derived HG-based features of $s$-closeness centralities furnish the proposed model’s robustness and resiliency. Specifically, the ML ensemble NIDS stumbled at initial simulation epochs but bounced back once it learned the failures (i.e., exhibiting resiliency) and then maintained the high-detection efficacy throughout the simulation (i.e., exhibiting robustness). As summarized in Table IV, numerical results suggest that the ML ensemble NIDS comprising $RF$ NIDS, HGI NIDS, and HGA NIDS along with Update-ALL-NIDS update rule (namely, the settings for case 5) exhibits robust potency in resolving the complex dynamics rendered by the combinations of malicious attacks, adversarial examples, attack threshold choices, and production environment.

VI. Conclusion

This work advances the use of a HG-based approach to devise a novel ML ensemble NIDS detecting malicious attacks and adversarial examples with high efficacy. Specifically, the work derives a set of $s$-closeness centrality metrics capturing the adversary’s signature pattern of port scan activities on a targeted machine. The work further investigates ways to incorporate these HG metrics to render features augmenting the NIDS training dataset to train the effective proposed ML ensemble NIDS.

Future research can extend these concepts from adding more HG metrics to using such metrics to analyze other types of network intrusions. Nontree-based NIDS models may reduce the size of the attack threshold to trigger and allow for more frequent and effective learning in the model retraining. In light of differences in the evading performance level of adversarial examples between commonly adopted ML models and generated adversarial examples, several other approaches to generate adversarial examples based on evolutionary computation and deep learning will be investigated further to examine adversarial robustness of the ML ensemble NIDS [8], [9]. Adversarial examples can also be used to examine the effectiveness of the HG-based ML ensemble NIDS adversarial robustness. Other techniques, such as incremental learning, could also be explored to help build a self-sustaining, intelligent NIDS.

ACKNOWLEDGMENT

The authors would like to thank two anonymous reviewers for their careful reading of this manuscript and their many insightful comments and suggestions on earlier versions of this article. The views expressed in this article are those of the authors and do not reflect the official policy or position of the National Intelligence University, United States Military.
REFERENCES

[1] M. J. De Lucia, P. Maxwell, N. D. Bastian, A. Swami, B. Jalaian, and N. Leslie, “Machine learning for raw network traffic detection,” in Proc. Artif. Intell. Mach. Learn. Multi-Domain Oper. Appl. (SPIE), 2021, pp. 185–194. [Online]. Available: https://arxiv.org/abs/2009.13250

[2] R. R. Devi and M. Abualkibash, “Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL KDD dataset—A review paper,” Int. J. Comput. Sci. Inf. Technol., vol. 11, no. 3, pp. 65–80, 2019.

[3] P. Maxwell, E. Alhajjar, and N. D. Bastian, “Intelligent feature engineering for cybersecurity,” in Proc. IEEE Int. Conf. Big Data, 2019, pp. 5005–5011.

[4] M. Chale and N. D. Bastian, “Challenges and opportunities for generative methods in the cyber domain,” in Proc. Winter Simul. Conf., 2021, pp. 1–12.

[5] D. A. Bierbrauer, M. J. De Lucia, K. Reddy, P. Maxwell, and N. D. Bastian, “Transfer learning for raw network traffic detection,” Expert Syst. Appl., vol. 211, Jan. 2023, Art. no. 118641.

[6] D. A. Bierbrauer, W. Kritzer, A. Chang, and N. D. Bastian, “Cybersecurity anomaly detection in adversarial environments,” in Proc. AAAI Fall Symp. Artif. Intell., 2021, pp. 1–7.

[7] S. M. Devine and N. D. Bastian, “An adversarial training based machine learning approach to malware classification under adversarial conditions,” in Proc. 54th Hawaii Int. Conf. Syst. Sci., 2021, pp. 827–836.

[8] E. Alhajjar, P. Maxwell, and N. Bastian, “Adversarial machine learning in network intrusion detection systems,” Expert Syst. Appl., vol. 186, pp. 1–13, Dec. 2021.

[9] M. Schneider, D. Aspinall, and N. D. Bastian, “Evaluating model robustness to adversarial samples in network intrusion detection,” in Proc. IEEE Int. Conf. Big Data, 2021, pp. 3343–3352.

[10] R. Bielawski, R. Gaynier, D. Ma, S. Lauzon, and A. Weimerskirch, “Cybersecurity of firmware updates,” U.S. Dept. Transp., Nat. Highway Traffic Safety Adm. Gov. Agency, Washington, DC, USA, Rep. DOT HS 812 807, 2020. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/cybersecurityoffirmwareupdatesdatesc2020.pdf

[11] M. Aljanabi, M. A. Ismail, and A. H. Ali, “Intrusion detection systems, issues, challenges, and needs,” Int. J. Comput. Intell. Syst., vol. 14, no. 1, pp. 560–571, 2021.

[12] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, and Y. Yao, “Towards network anomaly detection using graph embedding,” in Proc. 20th Int. Conf. Conf. Comput. Sci., 2020, pp. 156–169.

[13] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” Cybersecurity, vol. 2, no. 1, pp. 1–22, 2019. [Online]. Available: https://doi.org/10.1007/s42083-019-00038-7

[14] L. Wang, A. Liu, and S. Jajodia, “Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts,” Comput. Commun., vol. 29, no. 15, pp. 2917–2923, 2006.

[15] A. Guzzo, A. Pugliese, A. Rullo, and D. Saccà, “Intrusion detection with hypergraph-based attack models,” in Proc. 3rd Int. Workshop, Graph Struct. Knowl. Represent. Reason., 2014, pp. 58–73.

[16] M. R. Raman, K. Kannan, S. K. Pal, and V. S. Sriman, “Rough set-hypergraph-based feature selection approach for intrusion detection systems,” Defence Sci. J., vol. 66, no. 6, pp. 612–617, 2016.

[17] M. Raman, N. Somu, K. Kannan, R. Liscano and V. S. Sriman, “An efficient intrusion detection system based on hypergraph—Genetic algorithm for parameter optimization and feature selection in support vector machine,” Knowl. Based Sys., vol. 134, pp. 1–12, Oct. 2017.

[18] X. An, J. Su, X. Li, and F. Lin, “Hypergraph clustering model-based association analysis of DDoS attacks in fog computing intrusion detection system,” EURASIP J. Wireless Commun. Netw., vol. 2018, no. 1, pp. 1–9, 2018. [Online]. Available: https://doi.org/10.1186/s13638-018-2545-9

[19] C. Joslyn et al., “Hypergraph analytics of domain name system relationships,” in Proc. Int. Workshop Algorithms Models Web-Graph (WAW), 2020, pp. 1–15.

[20] A. Kalekar, N. Khatriya, S. Chakranarayan, and S. Wadkar, “Real time intrusion detection system using machine learning,” Int. J. Eng. Res. Technol., vol. 3, no. 2, pp. 185–187, 2014.

[21] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, “Practical real-time intrusion detection using machine learning approaches,” Comput. Commun., vol. 34, pp. 2227–2235, Dec. 2011.

[22] S. Thiriramanne, L. Jayawardana, L. Yasakethu, P. Liyanarachchi, and C. Hewage, “Deep neural network based real-time intrusion detection system,” SN Comput. Sci., vol. 3, p. 145, Jan. 2022.

[23] T. J. Shipp et al., “Advancing the research and development of assured artificial intelligence and machine learning capabilities,” 2020, arXiv:2009.13250.

[24] D. A. Bierbrauer, A. Chang, W. Kritzer, and N. D. Bastian, “Anomaly detection in cybersecurity: Unsupervised, graph-based and supervised learning methods in adversarial environments,” 2021, arXiv:2105.06742.

[25] Z. Kolter and A. Madry. “Adversarial robustness—Theory and practice,” 2019. [Online]. Available: https://adversarial-ml-tutorial.org/

[26] F. Stonedahl and U. Wilensky. “Netlogo virus on a network,” 2008. [Online]. Available: http://ccl.northwestern.edu/netlogo/models/VirusOnANetwork

[27] (Central Inf. Comm. Gov. Body, Munirka, Delhi). Canadian Institute for Cybersecurity (CIC)—Intrusion Detection Evaluation Dataset (CIC/IDS2017), 2017. [Online]. Available: https://www.unic.ca/cic/datasets/ids-2017.html

[28] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” Portugal, 2018. [Online]. Available: https://paperswithcode.com/dataset/cicids2017

[29] K. E. Monson, D. L. Arendt, S. G. Aksoy, B. L. Praggastis, E. Purvine, and C. A. Joslyn. “HypernetX,” 2019. [Online]. Available: https://www.pnnl.gov/copyright/hypernetx

[30] B. Praggastis et al., “HypernetX,” 2022. [Online]. Available: https://github.com/pnnl/HyperNetX

[31] S. G. Aksoy, C. Joslyn, C. O. Marrero, B. Praggastis, and E. Purvine, “Hypernetwork science via high-order hypergraph walks,” EPJ Data Sci., vol. 9, p. 16, Dec. 2020. [Online]. Available: https://doi.org/10.11007/s13688-020-00231-0

[32] M. Ali, “Pycare—Classification,” 2020. [Online]. Available: https://pycare.readthedocs.io/en/latest/api/classification.html

[33] M. I. Nicholae et al., “Adversarial robustness toolbox v1.0.0,” 2019, arXiv:1807.01069.

[34] S. Baluja and I. Fischer, “Adversarial transformation networks: Learning to generate adversarial examples,” 2017, arXiv:1703.09387.

Zong-Zhi Lin received the B.S. and M.S. degrees in industrial engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1988 and 1990, respectively, the Ph.D. degree in industrial and operations engineering in operations research concentration from The University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1999, and the M.S. degree in science and technology intelligence with dual concentrations in cyber intelligence and data science in intelligence from the National Intelligence University, Bethesda, MD, USA, in 2022. He is a Research Affiliate with the Intelligent Cyber-Systems and Analytics Research Laboratory, Army Cyber Institute, United States Military Academy, West Point, NY, USA. He is also a Senior Workforce/Operations Research Analyst with U.S. Department of Defense. He is interested in developing intelligent cyber-systems and analytics to enhance cyber-security protection, and researching methodologies characterizing complex systems to address people analytics applications.

Thomas D. Pike received the Ph.D. degree in computational social science under Prof. Robert Axtell from George Mason University, Fairfax, VA, USA, in 2019. He was recently the Dean of the Oettinger School of Science and Technology Intelligence, National Intelligence University, Bethesda, MD, USA. He is a Strategic Intelligence Officer with the United States Army. He has served in numerous intelligence and infantry positions in his military career from the platoon to the combatant command. He is a Strong Advocate for technical literacy and open-source software as a global public good. He is also one of the Lead Developers on Mesa (Agent-Based Models in Python). His main areas of research are on the use of agent-based models to support policy development and using complex systems to understand the dynamics of resilient and robust democracies.
Mark M. Bailey received the B.S. degree in chemical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2006, the first M.S. degree in chemical engineering from the University of Kansas, Lawrence, KS, USA, in 2008, the Ph.D. degree in bioengineering from The University of Kansas, Lawrence, KS, USA, in 2010, and the second M.S. degree in science and technology intelligence from the National Intelligence University, Bethesda, MD, USA, in 2018.

He is an Associate Professor with the National Intelligence University, where he is the Department Chair for Cyber Intelligence and Data Science as well as the Director of the Data Science Intelligence Center. He is interested in complex systems, AI safety, and their intersection with national security issues.

Nathaniel D. Bastian (Senior Member, IEEE) received the B.S. degree (Hons.) in engineering management (electrical engineering) from United States Military Academy (USMA), West Point, NY, USA, in 2008, the M.S. degree in econometrics and operations research from Maastricht University, Maastricht, The Netherlands, in 2009, and the M.Eng. degree in industrial engineering and the Ph.D. degree in industrial engineering and operations research from The Pennsylvania State University, University Park, PA, USA, in 2014 and 2016, respectively.

He is a Data and Decision Sciences Division Chief, a Senior Research Scientist, and the Chief Data Scientist of the Army Cyber Institute, Department of Electrical Engineering and Computer Science, USMA. He is also on the Faculty of Systems Engineering, Mathematical Sciences, and Computing, USMA. His research interests aim to develop innovative, assured, intelligent, human-aware, data-centric, and decision-driven capabilities for cyberspace operations.