ASSOCIATION FOR COMPUTING MACHINERY

DIGITAL THREATS: RESEARCH & PRACTICE

VOLUME 6 • NUMBER 1 • 2025 ARTICLES 1–6

Association for Computing Machinery

Digital Threats

RESEARCH & PRACTICE • 2025

continued on back cover

Open Access

dtrap.acm.org

ACM
Digital Threats: Research & Practice

ACM
1601 Broadway, 10th Floor
New York, NY 10019-7434
Tel.: (212) 869-7440
Fax: (212) 869-0481
https://www.acm.org

Home Page: https://dtrap.acm.org/

ACM Digital Threats: Research & Practice (e-ISSN: 2576-5337) is published quarterly in Winter, Spring, Summer, and Fall by the Association for Computing Machinery (ACM), 1601 Broadway, 10th Floor, New York, NY 10019-7434.

For manuscript submissions, subscription, and change of address information, see inside backcover.

Introduction to the Special Issue on Incident Response

Background

The beginnings of coordinated internet security can be traced back to the late 1960s and early 1970s, when the first computer networks were developed and then further connected to form the ARPANET. In the early days, there was a relatively low level of security and fewer concerns about malicious users. Not long after, computers and networks became increasingly widespread. There was an explosion in the growth of connected devices, access to systems, connected networks, critical services, and sensitive data all being exchanged through inter-connected networks (the internet). A clear need to protect sensitive information and prevent unauthorized access has emerged.

One of the earliest recorded internet-wide security events took place in 1986, when a computer virus called the "Morris Worm" disrupted the operation of thousands of computers. Many network administrators were unprepared to deal with computer programs or users that deliberately affected file integrity, guessed passwords, or exploited vulnerabilities to disrupt systems. After the event, the community began exchanging lessons learned and preparing for the new eventuality of malicious users. This event also demonstrated the need for a coordinated approach to internet cybersecurity incidents. In the aftermath of the attack, DARPA asked the Software Engineering Institute to establish a computer emergency response team, which has come to be known as CERT coordination center.

The internet is a different place today. Many organizations now have their own dedicated **Computer Security Incident Response Teams (CSIRT)** to help perform cybersecurity operations including managing, responding to cyber incidents, and helping mitigate the impact of cyberattacks. The cybersecurity threat environment facing organizations has also changed dramatically [1]. CSIRTs are often responsible for coordinating within their organization and with outside groups and agencies to share information and resources about the threats they are facing, the methods they are using to combat those threats, lessons learned, experiences shared, assistance requested, and much more.

Despite this growth, academic research in the field of Incident Response has been relatively limited. Challenges to the research include (1) a lack of access to open data for evaluation, (2) overcoming many hurdles (including non-disclosure agreements) in obtaining and using sensitive security data, and (3) task performance or information exchange occurring without researcher involvement (if it happens at all).

However, as cybersecurity incidents continue to accelerate in both speed and impact on organizations, there are growing calls for traditional and interdisciplinary research approaches to the study and improvement of cybersecurity Incident Response. Research has historically come from the field of computer science and cybersecurity, but we feel there is great potential for incorporating and applying perspectives from other fields as well including philosophy, psychology, human computer interaction, business operations, law, and more. Managing incidents is the responsibility of the whole organization, and more teams are incorporating talent from traditional

ACM Reference format:
Samuel Perl, Jeroen van der Ham-de Vos, Thomas Schreck. 2025. Introduction to the Special Issue on Incident Response. *Digit. Threat. Res. Pract.* 6, 1, Article 1 (March 2025), 3 pages.
https://doi.org/10.1145/3722125

ACM 2576-5337/2025/3-ART1
https://doi.org/10.1145/3722125

functions such as legal departments, businesses, supply chain management, and more into their CSIRT and cyber operational teams.

We have some great contributions in this special issue.

Unveiling Cyber Threat Actors: A Hybrid Deep Learning Approach for Behavior-based Attribution. Attribution is a common challenge in incident response and can help steer the response. Ertan et al. have applied natural language and machine learning tools to perform automatic attribution based on the behavioral patterns of these threat actors. They show the accuracy of their methodology on different types of datasets with different levels of information, showing some promising results.

Automated ATT&CK Technique Chaining. The MITRE ATT&CK framework serves as a valuable resource for classifying adversary behaviors across different attack stages. However, it does not inherently provide insights into what most likely occurred before or after a given technique used by attackers. Eian et al. enhance the framework by introducing semantic analysis and open source tools that can better link the most likely stages of an attack, allowing incident responders to hunt for the most likely adversary's tactics first. By integrating semantic modeling of expert knowledge with data-driven techniques trained on real-world security incidents, their approach helps answer critical questions. Given an observed attack behavior, what technical changes most likely preceded it? and what are the most likely next steps for the adversary? This advancement improves threat analysis, enabling security teams to respond more effectively to cyber incidents by better anticipating, responding, and mitigating potential threats in real time.

FedNIDS: A Federated Learning Framework for Packet-based Network Intrusion Detection System. Network intrusion detection systems have traditionally been an important tool in the toolbox of an incident responder. Recently, **Deep Learning (DL)** capabilities have been applied in this context to great success. However, the DL approach can only be applied in a centralized solution, or at least one where the data processing can be done centrally. Nguyen et al. propose a decentralized approach based on federated learning, which has the added advantage that it can be tuned differently for different segments of a network. Different individuals, groups, and departments in an organization will frequently have different network baselines. There is no "one size fits all" network baseline. Decentralized network analysis research and new models to perform traffic analysis intend to reduce the number of false positives for incident responders.

On Collaboration and Automation in the Context of Threat Detection and Response with Privacy-Preserving Features. Nitz et al. examine the challenges organizations face in cybersecurity defense, particularly in the context of privacy-preserving collaboration. Through interviews with professionals from eight different organizations, they explore how privacy concerns impact the adoption of collaborative threat intelligence sharing and response automation. Their study highlights both the potential benefits and the obstacles associated with implementing privacy-preserving techniques in threat detection and Incident Response. To address these challenges, they propose a reference architecture for secure data sharing, aimed at improving collaboration while maintaining organizational and user privacy. Their findings provide valuable insights by outlining the specific current limitations, offering a foundation for future research and the development of more effective, privacy-conscious cybersecurity strategies.

Building a Better SOC: Towards the Ontology for Security Operations Center Assistance and Replication (OSCAR). **Security Operations Centers (SOCs)** play a crucial role in cybersecurity, yet there is no unified framework that comprehensively defines the key components required to develop an effective SOC. While various tools, strategies, and methodologies exist, they often address individual aspects rather than providing a holistic approach. Novak et al. aim to bridge this gap by proposing the **Ontology for SOC Creation Assistance and Replication (OSCAR)**. This ontology is designed to systematically outline the critical people, processes, and technologies involved in SOC development. Leveraging data-driven insights from cybersecurity experts through interviews and surveys, the research captures and structures decision-making processes, identifying how constraints impact

SOC implementation. By formalizing this knowledge into an ontology, OSCAR provides a structured and replicable framework that enhances our understanding of SOCs, facilitates more effective discussions, and supports organizations in building, assessing, and optimizing their security operations.

Samuel Perl
Carnegie Mellon University, Pittsburgh, PA, USA

Jeroen van der Ham-de Vos
University of Twente, Enschede, The Netherlands

Thomas Schreck
HM Munich University of Applied Sciences, Munich, Germany

Guest Editors

Reference

[1] Robin Ruefle, Audrey Dorofee, David Mundie, Allen D. Householder, Michael Murray, Samuel J. Perl. 2014. Computer security incident response team development and evolution. *IEEE Security & Privacy* 12, 5 (2024), 16–26. DOI: https://doi.org/10.1109/MSP.2014.89

Unveiling Cyber Threat Actors: A Hybrid Deep Learning Approach for Behavior-Based Attribution

EMIRHAN BÖGE, Sabanci University, Istanbul, Türkiye
MURAT BILGEHAN ERTAN, Vrije Universiteit Amsterdam, Amsterdam, Netherlands
HALIT ALPTEKIN, PRODAFT, Yverdon-les-Bains, Switzerland
ORÇUN ÇETIN, Sabanci University, Istanbul, Türkiye

In this article, we leverage natural language processing and machine learning algorithms to profile threat actors based on their behavioral signatures to establish identification for soft attribution. Our unique dataset comprises various actors and the commands they have executed, with a significant proportion using the Cobalt Strike framework in August 2020–October 2022. We implemented a hybrid deep learning structure combining transformers and convolutional neural networks to benefit global and local contextual information within the sequence of commands, which provides a detailed view of the behavioral patterns of threat actors. We evaluated our hybrid architecture against pre-trained transformer-based models such as BERT, RoBERTa, SecureBERT, and DarkBERT with our high-count, medium-count, and low-count datasets. Hybrid architecture has achieved F1-score of 95.11% and an accuracy score of 95.13% on the high-count dataset, F1-score of 93.60% and accuracy score of 93.77% on the medium-count dataset, and F1-score of 88.95% and accuracy score of 89.25% on the low-count dataset. Our approach has the potential to substantially reduce the workload of incident response experts who are processing the collected cybersecurity data to identify patterns.

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies → Machine learning**; **Neural networks**; **Natural language processing**; **Artificial intelligence**;

Additional Key Words and Phrases: Digital forensics, threat intelligence, machine learning, threat actor attribution, deep learning, natural language processing

ACM Reference format:
Emirhan Böge, Murat Bilgehan Ertan, Halit Alptekin, and Orçun Çetin. 2025. Unveiling Cyber Threat Actors: A Hybrid Deep Learning Approach for Behavior-Based Attribution. *Digit. Threat. Res. Pract.* 6, 1, Article 2 (February 2025), 20 pages.
https://doi.org/10.1145/3676284

1 Introduction

Cyber threat actor (CTA) attribution, a critical aspect of **cyber threat intelligence (CTI)** and digital forensics, thus incident response research, is the process of identifying the responsible party or actor behind a particular cyber-attack. The significance of attribution lies in determining the origin of past attacks and enabling predictive

Emirhan Böge and Murat Bilgehan Ertan contributed equally to this research.

Authors' Contact Information: Emirhan Böge, Sabanci University, Istanbul, Türkiye; e-mail: emirhanboge@sabanciuniv.edu; Murat Bilgehan Ertan (corresponding author), Vrije Universiteit Amsterdam, Amsterdam, Netherlands; e-mail: m.b.ertan@student.vu.nl; Halit Alptekin, PRODAFT, Yverdon-les-Bains, Switzerland; e-mail: halit@prodaft.com; Orçun Çetin, Sabanci University, Istanbul, Türkiye; e-mail: orcun.cetin@sabanciuniv.edu.

ACM 2576-5337/2025/2-ART2
https://doi.org/10.1145/3676284

analyses for potential future attacks. One dimension of CTA attribution that enhances its accuracy and relevance is the incorporation of behavioral signatures and threat actor profiling.

Behavioral signatures refer to distinct patterns of activities that can be associated with a specific threat actor. On the other hand, threat actor profiling involves creating comprehensive profiles of cybercriminals based on a range of attributes, including but not limited to their behavioral signatures, tools used, **tactics, techniques, and procedures (TTPs)**. Profiling not only identifies the actors but also establishes definitive guidelines for attribution. Thus, a combination of behavioral signatures and threat actor profiling makes attribution more precise and actionable.

One of the primary problems with cybercriminal attribution and digital forensics is the quality of the threat intelligence data. Some analysts perceive many threat intelligence feeds to be of varying quality. They may sometimes lack richness, meaning they may not contain enough detailed information to establish reliable behavioral signatures or construct comprehensive profiles. Consequently, analysts must put in substantial effort to process and make sense of the collected intelligence often combing through sparse and noisy data to identify key indicators of attribution. Hence, integrating behavioral signatures and threat actor profiling could enrich the attribution process and make it more efficient and effective.

A crucial challenge in CTI research is the need for a professional and automated approach to assigning a threat to a specific actor or group, using the commands carried out by that threat actor as a basis. Digital forensics experts can significantly benefit from attributing threats to specific actors based on command analysis. By studying the unique commands and sequences a threat actor uses, experts can discern distinct behavioral patterns that act like digital fingerprints. This facilitates quicker and more accurate investigations by offering insights into the attacker identification. Previously, threat actor attribution was done by various methods, and none of this research utilized the sequence of commands executed by the threat actors. For instance, a study uses high-level indicators of compromise, such as TTPs and malware [17]. At the same time, some studies have also used CTI reports to attribute cybercriminals [10]. In this research, we refrain from being restrained by TTPs or high-level indicators as CTAs have developed new strategies to make it hard for analysts to detect them [25].

Attributing CTAs using the sequence of commands they have executed can provide valuable insight into their behavioral processes, which can ease the attribution process. However, detecting CTAs based on the commands they have executed is not a feasible task to be accomplished by most analysts. This process needs not only experience but also lots of workload and time. Therefore, there needs to be a study on CTA attribution based on the actions and commands of the threat actors. We aim to fill this gap by providing CTA attribution based on the sequence of commands they have executed. Our methodology emphasizes the importance of soft attribution, as it allows us to profile and categorize threat actors based on observed behaviors without needing the concrete evidence required for hard attribution. This approach is particularly effective in a landscape where direct evidence can be elusive, and attackers often use sophisticated methods to conceal their identities. Moreover, an article from Virus Bulletin draws attention to the method of *attack replication*, which is an advanced defense technique that consists of mimicking the attacker completely, including the command that the attacker has executed, and stating that this is one of the most powerful yet least adopted tool [5]. In our study, even though it is not based entirely on attack replication, we are leveraging the original commands that the attacker executes not the replica ones. Our approach empowers a more informed approach to decision-making, providing a valuable starting point and a strategic direction for the CTA attribution process. Organizations can adapt this research into their infrastructure to attribute the CTAs that they have encountered.

The dataset used in this study comprises unique identifiers of threat actors. Threat actors and CTA groups use various frameworks, commands, and malware to facilitate their activities. In our case, a high portion of the data consists of commands executed using the Cobalt Strike. The sequences of commands executed by CTAs vary significantly, with the volume of commands ranging from as many as 1,000 to as few as 15 per threat actor. This imbalance in dataset sizes across different CTAs may negatively affect the performance of the trained deep learning models due to the potential for both overfitting in data-rich scenarios and underfitting in data-sparse scenarios. To address this challenge and overcome its impact on model robustness and generalization, we partitioned the data into three distinct groups: high-count, medium-count, and low-count datasets. This categorization enables targeted training and validation approaches that accommodate the diverse data volumes, ensuring that our models are not biased towards the characteristics of more extensively represented data. The decision to create separate datasets instead of a single bucketed dataset, which is the low-count dataset, was strategic, aiming to ensure that our evaluation metrics genuinely reflect the model's ability to perform well under varied data conditions. By adopting this approach, we aim to enhance model performance across varied operational contexts, which reflects more realistic conditions that organizations might encounter. Further details on the methodology and its justification are provided in Section 5.3, emphasizing our choices to maintain integrity and reliability in model assessment.

Additionally, we have developed a standardization method to simplify the unprocessed commands into a **standardized command language (SCL)** for deep learning models, which significantly improved the performance of the model. This standardization process involves converting variations of commands, which might differ due to syntactic discrepancies or the unique ways in which different threat actors format their commands, into a consistent structure. Moreover, this standardization helps in mitigating issues like overfitting, where a model trained on highly specific command formats might perform poorly when exposed to slightly different commands in practice. By abstracting the commands to a more generalized form, the models can better generalize from the training data to real-world scenarios, where the exact wording of commands might vary, but the underlying intent or action remains the same.

In this study, we leverage **natural language processing (NLP)** architectures such as transformer [29], which learns from a sequence of tokens to train a machine learning model. These techniques learn from the positions of the tokens and can understand the underlying relationship between these tokens. Understanding the relationship between the sequence of commands and their relations benefits the attribution process as it uncovers their hidden features. Additionally, we have further included different deep learning [12] methods, merging the strengths of the transformer architecture with **convolutional neural networks (CNN)** [13]. This article presents a uniquely tailored hybrid deep learning architecture that integrates various methods specifically designed for the complex task of attributing threat actors. We compared our hybrid deep learning architecture against notable pre-trained models. Namely, **Bidirectional Encoder Representations from Transformers (BERT)** [4], RoBERTa [14], SecureBERT [1], and DarkBERT [11].

Our main contributions in this article are:

- We present the first study for analyzing and attributing the malicious commands executed by threat actors in compromised machines, which is usually an inefficient and time-consuming process that digital forensics or incident response specialists do.
- We have created a **Standardized Command Language Converter (SCLC)** that converts unprocessed commands into a standardized language to improve the efficiency and accuracy of NLP architectures and reduce overfitting.
- We develop a hybrid deep learning architecture for threat actor attribution, which outperforms existing pre-trained transformer-based models like BERT, RoBERTa, SecureBERT, and DarkBERT. The hybrid architecture achieved F1-score of 95.11% and accuracy score of 95.13% on the high-count dataset, F1-score of 93.60% and an accuracy score of 93.77% on the medium-count dataset, and F1-score of 88.95% and an accuracy score of 89.25% on the low-count dataset.[1]

This article is structured as follows: in Section 2, we mention the previous studies about the attribution process. Later in Section 3, we give background information about the techniques we have used. Section 4 explains the dataset, including its source and content. Section 5 explains the implementation methods for this study, which are the SCL component that processes our dataset, the proposed hybrid deep learning architecture, and the data

[1]https://github.com/bogertaNET/Unveiling-CTAs

preparation step for experimentation. Lastly, in Sections 6 and 7, we first show the results of our experiments, explain, discuss them, and point out some limitations.

2 Related Works

Traditionally, manual analysis is the most common approach for attribution [19]. As far as we are aware, our study is the first to propose a CTA attribution based on the *sequence of commands* that these actors have executed using various NLP and machine learning techniques. Nonetheless, there have been several attempts at threat actor attribution using machine learning and deep learning techniques based on the actor's behavior.

Recent studies proposed the idea of finding similarities between malware, hence, de-anonymizing the CTA based on the source code and the behavior of the malware [2, 20]. Moreover, Rosenblum et al. [23] unfold the idea of authorship attribution of program binaries using stylistic similarities of authors between programs using machine learning techniques.

Noor et al. [17] conducted a study utilizing NLP and machine learning techniques to analyze CTI reports. Their main goal was to identify and profile CTAs targeting FinTech based on the specific patterns of their attacks. To accomplish this, they applied an NLP technique known as distributional semantics. They trained and evaluated a variety of machine learning and deep learning models using these publicly available CTI reports. Remarkably, the deep learning model they developed achieved an accuracy of 94%. Irshad and Siddiqui [10] have also focused on attributing cyber-threat actors using NLP and machine learning techniques to analyze unstructured CTI reports. With cyber-attackers using obfuscation and deception to hide their identities, the researchers aimed to develop an automated system for extracting features from these reports to profile and attribute the attacks. These features include tactics, techniques, tools, malware, and target information. They use an embedding model called "Attack2vec," trained on domain-specific embeddings. Various machine learning algorithms, such as decision tree, random forest, and support vector machine, are utilized for classification. The model achieved an accuracy of 96%. Perry et al. [19] proposed a method for attack attribution that involves the textual analysis of threat intelligence reports, employing NLP and machine learning techniques. The researchers developed a unique text representation algorithm capable of capturing contextual information. Their approach utilizes vector space representation of incident reports obtained from a blend of labeled reports and an extensive corpus of security literature. Upon the previous study, a new study by Puzis and Angappan [24] emerged to attribute threat actors based on similar CTI reports. Unlike the traditional machine learning methods that focus on analyzing malware samples, this research proposes a deep learning architecture for the task of attribution [24]. The authors use the same dataset as Perry et al. used in their research [19].

Han et al. [6] developed a method to detect the unique characteristics of **Advanced Persistent Threats (APTs)**. These threats are notorious for their *low-and-slow* attack patterns. To identify these anomalies, they leveraged provenance graphs, which offer rich contextual and historical information [6]. Regarding APT detection, Milajerdi et al. [16] introduced HOLMES, yet another system for APT detection, by leveraging the correlation of suspicious information flows that arise during an attack. Moreover, while Han et al. and Milajerdi et al. focus primarily on detecting APT groups, our approach can detect any threat actor present within our dataset. Notably, we harness the power of NLP for our detection process, a technique not used by these studies.

There have also been various studies that mentioned the importance of replication of the attack. Guerrero-Saade presented the idea that attack replication enables extreme opportunities in defense and offense [5]. However, it is essential to mention that this study does not directly relate to the replication of the attack; instead, it focuses on the actual attack logs. For instance, the dataset used only consists of the executed commands in an attack, nothing more; because of that, our model can only learn which commands the actors execute and their style of execution. Guerrero-Saade also mentions some profiling characteristics that are helpful for actor attribution and actor profiling. While there is no strict guideline for profiling, and it consists of various aspects including but not limited to technical aspects and social aspects of the actor, it indeed includes the commands that are executed and the execution style of the commands [5] which is the backbone of this research, meaning, our model is already paying attention to the traits that are important to the profiling.

3 Background

This section of the article is dedicated to providing a foundation of the core concepts and a more comprehensive understanding of the context of our research while underlining the specific challenges and considerations within CTA attribution research.

3.1 Transformer Architecture

The transformer architecture was first proposed by Vaswani et al. [29]. They differ from previous NLP techniques as they are much more capable of handling long-range dependencies. Transformer uses an attention mechanism that helps the model create global dependencies between the input and output. This architecture is composed of two main components. Namely, an encoder and a decoder and both of these components have self-attention and position-wise feed-forward networks. The self-attention mechanism replaces the idea of recurrence, which was used primarily in the previous NLP architectures such as long short-term memory [9]. This mechanism of the transformer architecture allows the developed model to address the problem of long-range dependencies by weighing the importance of different inputs for generating each output in the sequence. In addition, a positional encoding mechanism was used in the transformer architecture because, unlike previous NLP architectures, transformers do not have a recurrent mechanism that enables the architecture to take the order of the sequence into account, which is a crucial aspect of NLP tasks. The positional encoding provides the sequence information to the transformer architecture by adding a unique signal to each input token, representing its position in the sequence. This allows the architecture to learn and generalize about positional relationships between tokens.

In this study, transformers can help detect CTAs based on their commands used by attackers. Command sequences that actors have executed often contain long dependencies where the meaning of a command might be influenced by a command executed long ago. Therefore, transformer architecture is used in our hybrid deep learning architecture. The transformer architecture has also been used for training larger language models such as BERT [4], RoBERTa [14], SecureBERT [1], and DarkBERT [11]. These models have been used as a baseline for specific tasks as they have been trained on a large corpus of texts. Using these pre-trained transformer-based models in a specific task requires fine-tuning the model with the new dataset for that specific task's domain. In our case, this dataset is the CTA's command sequences.

3.2 CNNs

CNNs [13] are a deep learning algorithm primarily used for processing structured grid data. CNNs have been most influential in image and video recognition applications. The core component of this architecture is the convolution operation, which allows the model to learn spatial hierarchies of features. For instance, features in the early layers can be edges and textures, and in late layers, features can be parts of objects.

To analyze CTA command sequences, we employ **one-dimensional CNNs (Conv1D)**. This approach captures local and contextual features from command sequences by sliding filters across the text. Utilizing varying filter sizes, which correspond to different n-gram analyses, the network becomes more able to extract diverse semantic and syntactic features that are indicative of threat actor behaviors. In our case, a CTA's commands can be considered text. Therefore, these sequences may have patterns that can give insights into detecting characteristics of the behavior of a threat actor, and 1D CNNs can extract these patterns.

3.3 Residual Connections

Residual connections, introduced by He et al. [7], are implemented by allowing the output of an earlier layer to skip some layers and be added to the output of a later layer. This is valuable because it mitigates the "vanishing gradient" problem that usually arises while training complex models. The vanishing gradient problem refers to gradients becoming extremely small in deeper layers of the trained model, which can slow down the training process. Residual connections alleviate this problem by enabling gradients to flow through from earlier layers to later layers of the model, improving the effectiveness of training in even very deep networks.

3.4 Regularization Techniques

Overfitting is a common problem that must be addressed, which happens when a model learns the training data too well and performs poorly on the unseen data. This can be mitigated with regularization techniques such as dropout [26] and weight decay [15].

Dropout is a regularization technique that randomly sets a fraction of input units to zero, which helps prevent overfitting, leading to a general and robust model. Moreover, spatial dropout [27] is a specific type of dropout used in CNNs that preserves the spatial coherency of the input data, which allows the model to learn the spatially local features. In our case, spatial dropout can be beneficial as it is much more capable of helping the architecture capture meaningful localized features. For example, the pixels in an image may form meaningful patterns in a local context, which can be edges, textures, and shapes. Similarly, local patterns may have significant meaning in command sequences, such as command flags, arguments, and their sequential order. Additionally, weight decay is another technique to mitigate the overfitting problem. This method prevents weights of the model from reaching large values that may lead to overfitting by adding a penalty term to the loss function.

3.5 Hyperparameter Tuning

Hyperparameters are parameters set before training the model that alter how the model works. For instance, they include dropout fraction, weight decay, learning rate, number of epochs that the model has to be trained, and many others. The process of deciding hyperparameters is crucial as it affects the model significantly.

3.6 Cross Validation

Cross validation technique is used for assessing the effectiveness and robustness of a model. First, we partition the dataset into k subsets. One of these subsets is used for validation, and the rest $k-1$ subsets are used for training the model. This process is repeated k times, each time updating the subset used as the validation set, and the training set as the rest. Finally, k results are averaged to produce a more robust metric. In addition, the standard deviation of the performance across the k folds is calculated, where a low standard deviation indicates that the model is robust and consistent across different subsets of data.

3.7 Metrics for Evaluation

We leverage two metrics for evaluating our model: accuracy and weighted F1-score. Accuracy simply measures the proportion of correctly classified instances.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}. \tag{1}$$

F1-score is a useful metric when dealing with imbalanced classes. It is the harmonic mean of precision and recall, which are calculated based on true positives, false positives, and false negatives.

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2}$$

For multi-class problems like ours with more than 2 CTAs, we calculate weighted average of F1-scores.

$$Weighted\ F1\ score = \frac{1}{N} \sum_{i=1}^{N} w_i \cdot F1\ score_i. \tag{3}$$

Table 1. Command Data

Field	**Type**	**Example**
data	string	execute_cmd: net group
timestamp	UNIX Epoch in seconds	1646158272

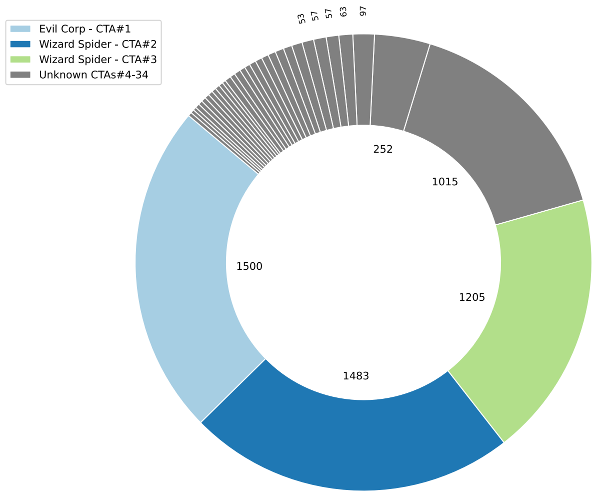


Fig. 1. Commands per CTA.

4 Dataset

European Threat Intelligence company ***Proactive Defense Against Future Threats (PRODAFT)*** provided private threat intelligence data collected from August 2020 to October 2022. Commands executed by the attackers were captured by *PRODAFT* using various private honeypot servers located in China, Europe, and the United States. The aim is to lessen the workload of cybersecurity analysts and facilitate progress in the field of CTI.

Dataset contains several attack frameworks, such as the Cobalt Strike. The raw form of the data consists of lists of commands executed by CTAs along with the exact time that is executed. The data is already clustered with CTA's unique identifications, such that the data are labeled with CTA's aliases. For example, Table 1 shows a sample command structure and its content. The command execution JavaScript Object Notation data consists of two fields, namely *data* and *timestamp*. The *data* field contains the information and commands executed by CTAs. On the other hand, the *timestamp* field is designed to hold UNIX epoch timestamp data.

The dataset mainly contains 34 CTAs with over 5,000 commands. The number of command sequences that the dataset contains for each CTA is not uniform, meaning while CTA#1 has 1,500 commands, CTA#25 only has 21 command sequences, as Figure 1 displays. Only one uses a custom malware, called *SocGholish*, and the rest of the threat actors use the Cobalt Strike to execute commands. The variety of malware is intended since we would like to demonstrate that our architecture can predict threat actors that use different malware and can also predict distinct actors even though they use the same malware. The top three CTAs in our dataset are attributed to various known threat actor groups. Their details can be seen in Table 2; affiliation of the rest of the CTAs is unknown, yet it is known that they are distinct.

More importantly, *PRODAFT* stated with strong confidence that one of the threat actors is attributed to the *SilverFish* group. This group has strong ties with the notorious SolarWinds incident and the globally recognized *Evil Corp* group [22]. Moreover, as the Health Sector Cybersecurity Coordination Center (HC3) of the U.S. claims,

Table 2. Details of the Important CTAs

Threat Actor	**# of Command Sequences**	**Malware**
Evil Corp—CTA#1	1500	SocGholish
Wizard Spider—CTA#2	1483	Cobalt Strike
Wizard Spider—CTA#3	1205	Cobalt Strike

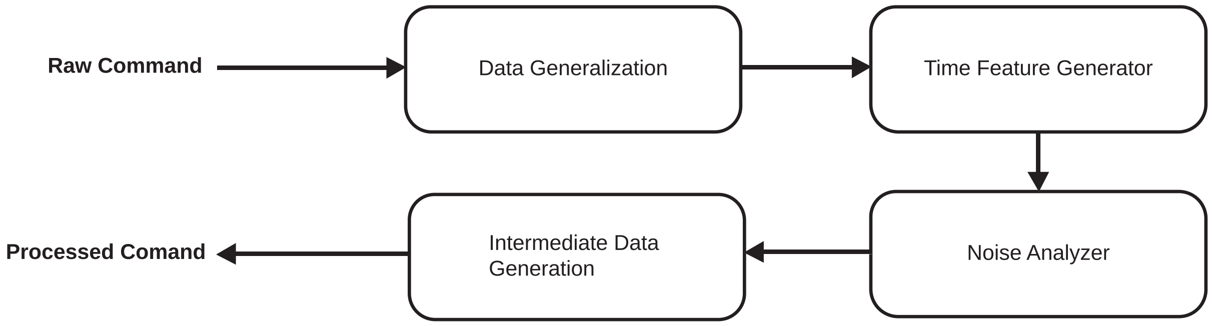


Fig. 2. SCLC schema.

Evil Corp is one of the most sophisticated cybercriminal syndicates in which they developed and operated several critical malware and ransomware variants, such as the *Dridex* malware that caused more than $100 million in theft [8]. In addition, two of the threat actors are also a subgroup of a threat actor group called *Wizard Spider* [21], and one of them is affiliated with group *Royal Ransomware* [3].

5 Methodology

5.1 SCL

Overfitting is a phenomenon used to describe when a developed machine learning model performs well on the dataset used to train the model but performs poorly on a dataset that the model has not seen before. For example, cybercriminals may use extremely specific file names, IP addresses, URLs, and dates. This can cause our machine learning model to learn those particular features and not learn the features that represent the behavioral and decision-making processes of the threat actor. The commands must be transformed into a more generic form to prevent this. Therefore, in this study, an intermediate component is created for changing each command into a more generic form to prevent the problem of lack of generality.

The SCLC aims to simplify unprocessed commands into an SCL tailored explicitly to standardize the command data. The SCLC entails transforming unstructured and diverse command data into a universal format that adheres to specific syntax and semantics defined by SCL, for instance, converting different file path syntax into a universal one. Creating such a language provides a more accurate, scalable, and uniform environment for language processing, increasing our NLP architecture's overall efficiency and accuracy. As shown in Figure 2, a raw command must be passed through four steps before it becomes a processed command.

5.1.1 SCLC—Data Generalization Step. The first place that SCLC is processing raw data. The motivation behind this step is to create a generic command mapping since different malware that CTAs use work in different environments and in nearly completely distinct manners. Therefore, we have manually inspected common and similar commands to create a mapping for similar commands into a single command. Moreover, all file paths, domains, usernames, and indications for a CTA are being obfuscated by SCLC.

Similar commands such as *execute_cmd*, *run*, and *start*, which have the same meaning, are mapped into the command *execute_cmd*. By doing so, our NLP architecture becomes more aware of the context, becoming

Table 3. Before SCL Generation

Field	Command
data	list jobs
when	1663713006
data	shell wmic \node:192.168.0.254 process call create "C:\ProgramData\sys.exe"
when	1663713268

Table 4. After Standardized Command Generation

Field	Command
data	list jobs
data	wait: zero hours four minutes twenty-one seconds
data	execute_cmd wmic \node:IPADDRESS process call create FPATH

independent of the malware that CTA uses. For instance, two threat actors who use different malware to execute the same command with different syntax, not necessarily, but probably. However, our NLP architecture should not distinguish CTAs by the malware they use but by their unique operating style.

5.1.2 SCLC—Time Feature Generation Step. The data includes the timestamp a command has executed, so one can easily deduce the seconds between two consecutive commands. Then, we calculate the elapsed seconds for each consecutive command and insert it as a command between them. Hence, we are imitating the wait time between words an author writes in their book, or our case, the thought process of a threat actor before executing any command.

5.1.3 SCLC—Noise Analyze Step. In this step, any command execution series with over 32 commands are separated into sequences with 32 commands. The main reason behind the split is to prepare a uniform execution series structure, in other words, to handle the unbalanced command distribution over the command execution series. In addition, without any splitting, the model's training becomes unfeasible, and the consequences of splitting are negligible.

5.1.4 SCLC—Standardized Data Generation Step. In the final step, nearly processed data is prepared; one can find more detailed information in Step 5.3. The main objective of this step is to double-check the data for duplicates and noises. For instance, even though a command execution series has only 32 commands, if it has more than 256 sequences, which is mentioned in Step 5.3, the data point is still dropped from the dataset to prevent noise. As a result, the intermediate data has been generated and is ready to be prepared for the model.

Sample input and output for the execution of the SCLC can be seen in Tables 3 and 4. As mentioned in *Data Generalization Step*, to create a uniform structure, all file paths are obfuscated into the text "FPATH," and all IP addresses are obfuscated into "IPADDRESS." Finally, in *Time Feature Generation Step*, a new command is inserted between two consecutive commands that specify the wait time between commands. Hence, unprocessed data has successfully inherited a new syntax and semantics.

5.2 Architecture

In this study, we use a hybrid deep learning structure that leverages the power of transformers and CNNs to attribute CTAs. Our architecture is designed to capture global and local contextual information within the sequence of commands. This is crucial because the significance of specific terms in the commands can drastically change depending on the broader, global context. For instance, a particular command used in one context may not help us to attribute the CTA. Still, when the relation between other commands is examined, this specific command can indicate the unique operational patterns of a particular CTA. Such understanding requires a hybrid

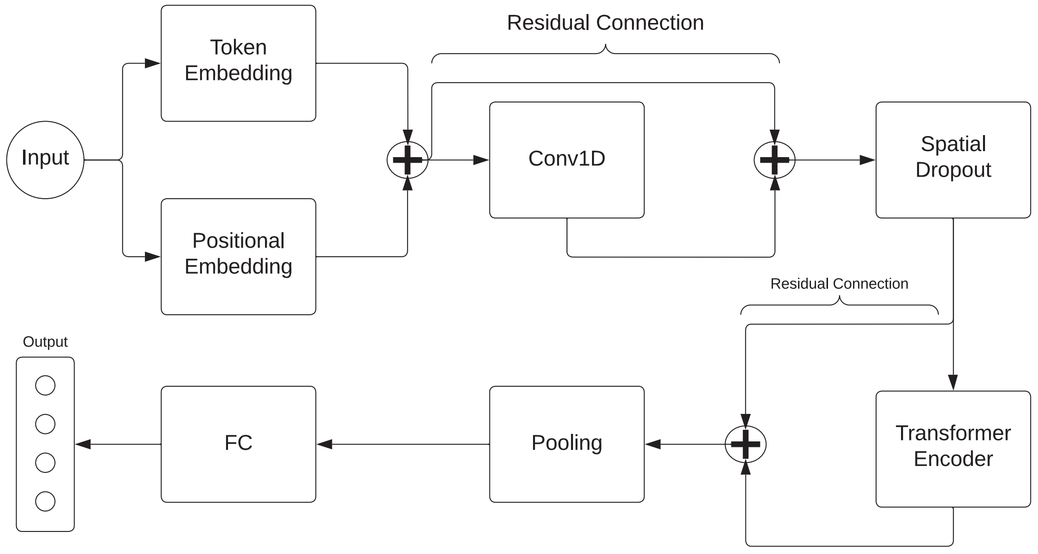


Fig. 3. Hybrid Architecture.

architecture that can comprehend both the immediate surroundings of a term, local context, and the broader sequence of commands, global context.

Our hybrid architecture consists of several key components:

— *Token and Positional Embedding*: These two separate embedding layers are used for encoding the sequence of input tokens and their positions in the sequence. These embedding layers enable our model to capture both the semantic meaning of the tokens and the order in which these tokens are represented in the sequence. The size of the embeddings is controlled by a parameter named hidden size, which is tuned to its best optimal size in the hyperparameter tuning experiments.
— *Conv1D*: This layer helps our model to capture local features from the input sequence. The 1D convolutional layer is used because our input data is in a sequential form, represented in 1D vectors. It uses a kernel that moves across the sequence to learn and extract meaningful patterns. The size of the kernel is determined by a parameter named kernel size, and the number of output filters is controlled by a parameter named number of filters. Again, these parameters are tuned to their best size in the hyperparameter tuning experiments.
— *Spatial Dropout*: This layer prevents overfitting. We have used spatial dropout instead of regular dropout as it preserves the spatial coherence of the input data and, as a result, allows a more efficient way of learning spatially local features. The dropout rate is determined in the hyperparameter tuning experiments.
— *Transformer Encoder*: This component captures the global dependencies in the command sequence. It consists of multiple transformer encoder layers with self-attention mechanisms. The parameter number of layers determines the number of layers, and the number of attention heads is controlled by the number of heads, as done in the other components; these parameters are tuned in the hyperparameter tuning experiments.
— *Residual Connections*: These connections are used after the Conv1D and transformer encoder layers. This component of our hybrid architecture enables the retention of important information throughout the deeper layers.

As seen in Figure 3, the input command sequence first goes through the token and positional embedding layers. Later, for local feature extraction, the output of these embedding layers is passed through a Conv1D layer with a residual connection. After applying spatial dropout, the sequence is fed into the transformer encoder, providing

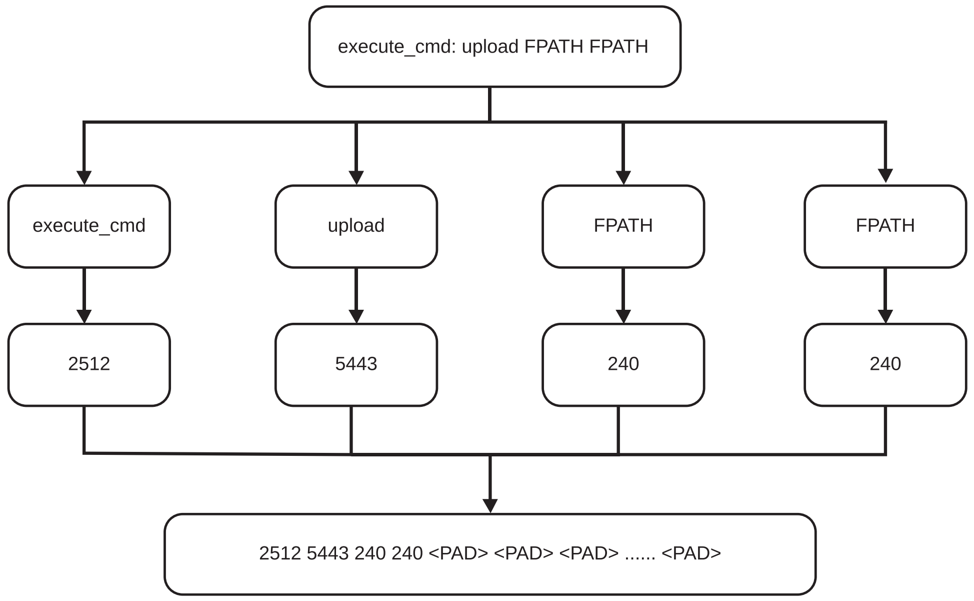

execute_cmd: upload FPATH FPATH
execute_cmd
upload
FPATH
FPATH
2512
5443
240
240
2512 5443 240 240 <PAD> <PAD> <PAD> ...... <PAD>

Fig. 4. Tokenization of commands.

an enriched new sequence with contextual information from all other positions in the sequence. Another residual connection is used at this stage. Subsequently, a mean pooling across the output is used for reducing the size of the output tensor, which concentrates the sequence's most crucial information. Finally, the pooled output from this layer is fed into a linear fully connected layer, which performs the final transformation of the data, giving us the final predictions.

5.3 Data Preparation

We transform the given sequence of commands into numerical tokens to enable our NLP architectures to understand and learn from the commands. As one can see from Figure 4, every unique word in commands is extracted and mapped into a number, and then we replace every word in the processed data with the corresponding number. For example, the command in Figure 4 shows that the command *execute_cmd* is interpreted as the number 2512. In addition, to make all of our sequences the same length, we are fixing the maximum length of a sequence into an optimal number of tokens, in our case, 256, and pad the shorter sequences to the maximum length by appending arbitrary tokens. After the tokenization is completed, our data is wholly represented numerically.

The chosen sequence length of 256 was found to offer the best balance between capturing sufficient contextual information and maintaining feasible computation times and memory usage. Moreover, choosing a shorter sequence length than 256 leads to insufficient context for making accurate predictions, as this may cause vital command elements to be trimmed. Additionally, longer sequence lengths require more padding to more CTA command data, which thins out the information present in the training data.

To reduce the impact of dataset size variability among the 34 threat actor datasets on model performance, we partitioned the data into three categories: high-count, medium-count, and low-count. This categorization serves several purposes. Firstly, it allows a more in-depth analysis of model performance across datasets of different sizes, which is critical in understanding the model's robustness and generalizability. High-count CTAs, having over 1,000 sequences each, provide a dense data environment where models can learn complex patterns without overfitting. Medium-count CTAs with more than 50 sequences, which includes also the CTAs in the high-count dataset as they have more than 50 sequences too, represent a more typical real-world scenario where there is enough data to train on but not as much as CTAs present in high-count datasets. Low-count CTAs include all 34 datasets, again including both high-count and medium-count CTAs, to examine the model's behavior in the most challenging scenario where data scarcity might disrupt the model's ability to learn effectively.

The decision to include all CTAs in the low-count category and to allow for overlapping between the low-count, medium-count, and high-count datasets was made to ensure comprehensive coverage and robust evaluation of our models across the entire spectrum of data availability. Additionally, the choice to combine CTAs with varying numbers of commands reflects the challenging environments that organizations often encounter, where they must manage a few datasets with large volumes of data alongside many smaller, sparser datasets. Thus, we have integrated low-count with medium-count and high-count and medium-count with high-count datasets. By evaluating models across these overlapping categories, we can more accurately assess their performance and understand the effects of data volume on learning outcomes. This approach ensures that our analysis extends beyond ideal conditions with plentiful data, which is crucial for developing universally robust and effective models.

- High-count: CTAs with more than 1,000 sequences (4 CTAs)
- Medium-count: CTAs with more than 50 sequences (10 CTAs)
- Low-count: All CTAs (34 CTAs)

Lastly, we split each dataset into training, validation, and test sets. The training set, which comprises 70% of the data, is used for training the model. The validation set makes up 15% of the data and is used to determine the best set of hyperparameters. Lastly, the remaining 15% of the data is the test set, which is used to evaluate the model's performance.

6 Experiments

6.1 Experimental Environment

To construct our experimental environment, we leveraged the deep learning library PyTorch [18]. In addition, our study utilized several pre-trained transformer-based models, specifically BERT, RoBERTa, SecureBERT, and DarkBERT. These models are implemented using the Hugging Face's[2] Transformers library.

Our dataset consists of command sequences of 34 threat actors. However, the CTAs are not uniformly sized. To deal with the imbalance in dataset sizes, we partitioned the CTAs into three categories: high-count, medium-count, and low-count. Experiments are conducted across these differently sized datasets using all of the aforementioned deep learning models to ensure a comprehensive evaluation. Later, as explained before, the dataset is divided into a training set, constituting 70% of the data, a validation set comprising 15%, and a test set, making up the remaining 15%.

6.2 Hyperparameter Tuning

For pre-trained transformer models, the tuned hyperparameters are the learning rate, the number of epochs the model is trained, and weight decay. Moreover, for our hybrid architecture, dropout rate, the number of epochs, hidden size, kernel size, **learning rate (LR)**, filters, heads, and layers are the hyperparameters tuned in the hyperparameter tuning phase.

We employed an extensive hyperparameter tuning process to optimize the performance of our models, and the best combination of hyperparameters was chosen considering the validation F1-score. The best set of hyperparameters and its metrics for the pre-trained transformer-based models can be seen in Table 5. Again, the best hyperparameters and their metrics for the hybrid architecture can be seen in Table 6. The change of performance across different dataset settings that can be observed in the aforementioned tables, which hints that as the number

[2]https://huggingface.co/

Table 5. Hyperparameter Tuning Results—Pre-Trained Transformers

AQ15

Model	**Dataset**	**LR**	**Epochs**	**Weight Decay**	**Val. Accuracy**	**Val. F1-score**
BERT	High-count	3e-05	6	0.05	0.9243	0.9241
BERT	Medium-count	3e-05	4	0.01	0.9042	0.9018
BERT	Low-count	3e-05	6	0.05	0.8602	0.8456
DarkBERT	High-count	3e-05	6	0.05	0.9307	0.9303
DarkBERT	Medium-count	3e-05	6	0.05	0.8881	0.886
DarkBERT	Low-count	2e-05	6	0.01	0.8143	0.7914
RoBERTa	High-count	3e-05	6	0.01	0.9256	0.9253
RoBERTa	Medium-count	3e-05	6	0.05	0.8731	0.8713
RoBERTa	Low-count	3e-05	6	0.05	0.8060	0.7818
SecureBERT	High-count	3e-05	6	0.05	0.9269	0.9268
SecureBERT	Medium-count	3e-05	6	0.05	0.8892	0.8880
SecureBERT	Low-count	3e-05	6	0.01	0.7987	0.7739

Table 6. Hyperparameter Tuning Results—Hybrid Architecture

Dataset	**Dropout**	**Epochs**	**Hidden Size**	**Kernel Size**	**LR**	**Filters**	**Heads**	**Layers**	**Val. Acc.**	**Val. F1**
High-count	0.1	100	128	3	0.002	128	4	4	0.9628	0.9626
Medium-count	0.1	100	128	5	0.02	128	16	2	0.9365	0.9350
Low-count	0.1	150	128	5	0.02	128	16	3	0.9103	0.9074

of commands per CTA in the available dataset decreases for training, the models generally show a decline in validation accuracy and F1-score. Compared to pre-trained models, hybrid architecture with the best set of hyperparameters performs generally well across all dataset settings.

6.3 Cross Validation

After the hyperparameter tuning process, the optimal hyperparameters for each model with each dataset setting are found. With the best hyperparameters set, each model is cross validated using a 10-fold cross validation method, which is explained in the background Section 3. Moreover, in this phase of the experimentation, training, and validation datasets are merged, leading to a dataset for cross validation constituting 85% of the dataset. The training and validation of the deep learning models in the cross validation step is done with this merged dataset.

Employing 10-fold cross validation presents us a more reliable performance estimation and shows the robustness of each model. This process minimizes the risk of the training process becoming skewed toward a particular division of the dataset. Additionally, by providing mean and standard deviation metrics, we give a comprehensive view of each model's strengths and weaknesses. The mean and standard deviation of the performance metrics, computed across the 10 folds, are summarized in Table 7.

When the results of the 10-fold cross validation are inspected, the hybrid architecture in all dataset settings consistently outperforms each pre-trained model. Additionally, the standard deviation of the F1 scores are generally low on all datasets for the hybrid architecture, which again showcases the robustness of the architecture. Moreover, it should be noted that SecureBERT performed comparably well on the medium-count dataset. Generally, the standard deviations are observed to be low, suggesting that the hybrid architecture, and the fine-tuned pre-trained models are relatively stable across different folds of the dataset. This stability is especially important in practical applications where the trained deep learning model has to generalize and perform well in unseen data.

Table 7. Mean and Standard Deviation Results in 10-Fold Cross Validation

Model	**Dataset**	**Mean Accuracy**	**Standard Deviation Accuracy**	**Mean F1**	**Standard Deviation F1**
BERT	High-count	0.9197	0.0141	0.9195	0.0142
BERT	Medium-count	0.8811	0.0167	0.8763	0.0175
BERT	Low-count	0.8501	0.0107	0.8318	0.0119
DarkBERT	High-count	0.9179	0.0143	0.9177	0.0143
DarkBERT	Medium-count	0.8994	0.0153	0.8975	0.0150
DarkBERT	Low-count	0.8096	0.0189	0.7834	0.0207
RoBERTa	High-count	0.9151	0.0145	0.9147	0.0144
RoBERTa	Medium-count	0.8964	0.0153	0.8943	0.0160
RoBERTa	Low-count	0.8264	0.0188	0.8049	0.0199
SecureBERT	High-count	0.9133	0.0148	0.9132	0.0151
SecureBERT	Medium-count	0.9000	0.0098	0.8990	0.0100
SecureBERT	Low-count	0.8071	0.0246	0.7799	0.0286
HybridArchitecture	High-count	0.9418	0.0087	0.9416	0.0090
HybridArchitecture	Medium-count	0.9161	0.0129	0.9157	0.0129
HybridArchitecture	Low-count	0.8855	0.0118	0.8814	0.0131

Table 8. Experiment Results

Model	**Dataset**	**Test Accuracy**	**Test F1**
Hybrid	High-count	**0.9513**	**0.9511**
Hybrid	Medium-count	**0.9377**	**0.9360**
Hybrid	Low-count	**0.8925**	**0.8895**
BERT	High-count	0.9129	0.9126
BERT	Medium-count	0.8824	0.8802
BERT	Low-count	0.8696	0.8569
DarkBERT	High-count	0.9167	0.9165
DarkBERT	Medium-count	0.8986	0.8975
DarkBERT	Low-count	0.8237	0.7998
RoBERTa	High-count	0.9180	0.9178
RoBERTa	Medium-count	0.9135	0.9127
RoBERTa	Low-count	0.8613	0.8487
SecureBERT	High-count	0.9206	0.9204
SecureBERT	Medium-count	0.9228	0.9220
SecureBERT	Low-count	0.8206	0.7972

The highest accuracy and F1 values are highlighted in bold.

7 Results

In this section, we delve into the outcomes obtained from the comprehensive evaluation of our hybrid deep learning architecture. The performance is compared against several established pre-trained transformer-based models, namely BERT, RoBERTa, SecureBERT, and DarkBERT. This juxtaposition intends to fully encapsulate the strengths and potential areas of improvement for each model and, importantly, highlight the performance of the hybrid architecture. In the final testing step, for training, the merged train and validation datasets are leveraged, which is 85% of the dataset, and for testing the test set is used, which consists of 15% of the dataset.

The results in Table 8 depict the success of the hybrid architecture's performance on the test set over all dataset settings after various experimental iterations. The extensive experimentation, including numerous runs of hyperparameter tuning and 10-fold cross validation, which ensures the robustness of each model, resulted

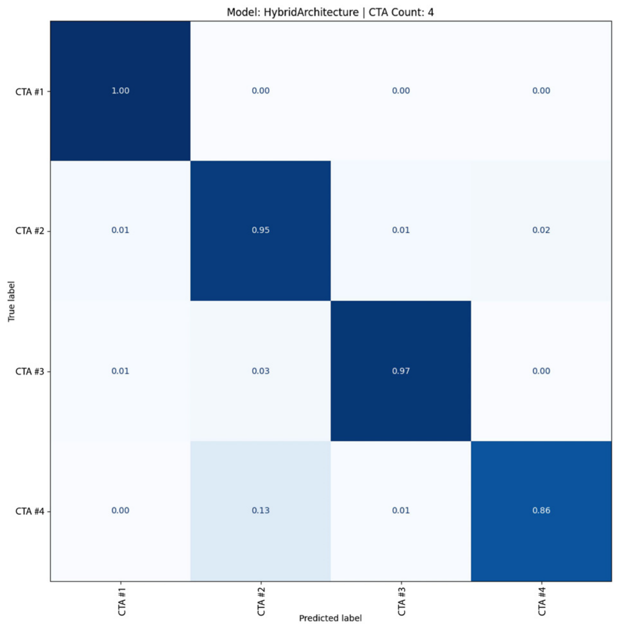


Fig. 5. High-count dataset.

in optimal performance settings for each model. Later, with the test set, the models were evaluated using test accuracy and F1 metrics, providing a holistic view of their performance across all classes of our multi-class classification problem. These results suggest that our hybrid deep learning architecture is the most effective model among the other pre-trained models that have been tested in terms of both accuracy and F1-score.

The confusion matrices for different dataset settings offer valuable insights into the model's performance by displaying the detection accuracy of the architecture for each CTA. Confusion matrices for hybrid architecture are depicted in Figures 5, 6, and 7 for high-count, medium-count, and low-count datasets, respectively. It should be noted that, the number of command sequences for each CTA is in sorted order. To explain, CTAs with higher indexes have much less sequence of commands compared to the lower index ones, for example CTA#1 has the most number of commands.

As shown in Figure 5 for the high-count dataset, the hybrid architecture demonstrates good performance across almost all CTAs. The high values along the diagonal indicate that the model is effectively classifying instances for each class. In Figure 6 for the medium-count dataset, a relatively balanced performance is observed. While the diagonal elements are strong, indicating good classification, there is room for improvement, for example, with the CTA #6. The low-count dataset, depicted in Figure 7 showcases a more challenging scenario. Although the model performs well for the majority of CTAs, it is observed that the model struggles with CTAs with lesser sequences of commands.

As can be seen from the confusion matrices, conducted experiments on different sizes of datasets indicate a correlation between the number of command sequences in each CTA and the performance of the deep learning models. Specifically, we observe:

— *High-count dataset*: The models achieve significantly better results, with higher accuracy and F1-scores. The large dataset size allows the models to learn complex patterns and generalize effectively.
— *Medium-count dataset*: A moderate decrease in performance metrics is observed, the models suffer from some limitations in capturing the behavior of the CTAs with lower examples of command sequences.

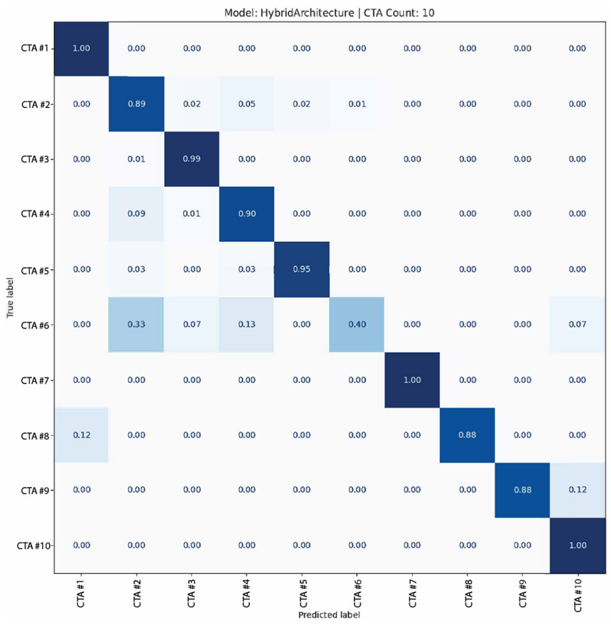


Fig. 6. Medium-count dataset.

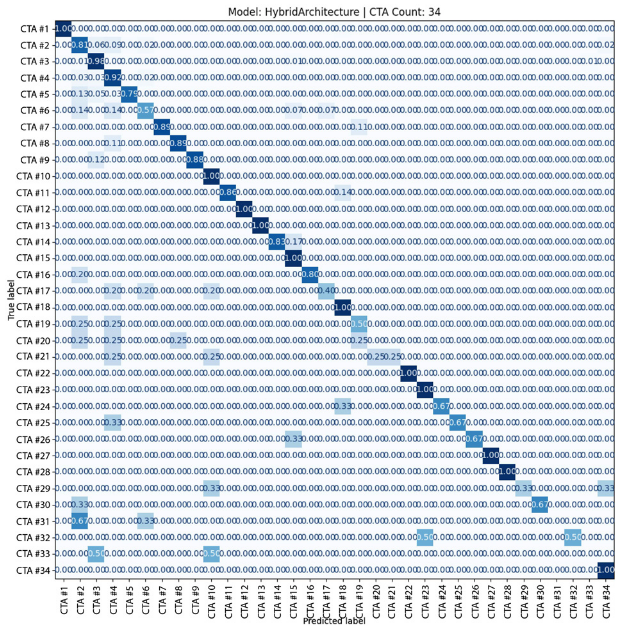


Fig. 7. Low-count dataset.

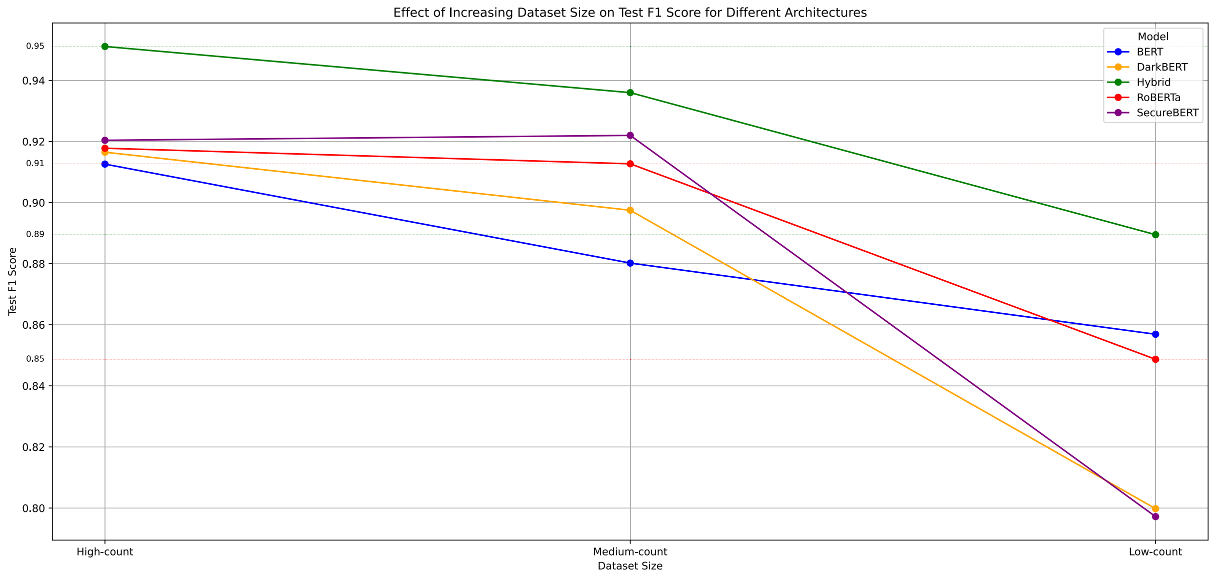


Fig. 8. Performance metrics for different dataset sizes.

— *Low-count dataset*: The models result in poorer performance, including lower accuracy and F1-scores. This is likely due to insufficient data, which prevents the model from learning the patterns in the CTAs with lower examples of command sequences well.

Figure 8 displays the relationship between the dataset sizes and the performance metrics. These findings exemplify the importance of dataset size for an effective model that detects CTAs. Nevertheless, it should be noted that the hybrid architecture proves to be relatively well across all dataset settings. This shows the architecture's ability to learn essential features of CTAs from their sequences of commands and generalize better in data-scarce environments. This resilience to data sparsity makes the hybrid architecture much more important as labeled datasets are often difficult to obtain in real-world applications.

8 Limitations and Future Work

However, it is important to acknowledge that, in cyberspace, actors continuously change their style to hide their signatures and characteristics, or they even borrow open-source frameworks to *hide in the noise* [5]. It is still unclear if this model can predict with high accuracy if *Evil Corp* switches to another malware, but there is still much to delve into in this study area. It is also essential to know that the CTA attribution process is not solely dependent on the sequence of the commands that the actor executes. There are numerous traits and methods that should be considered during the attribution, meaning that our model should not be the only source for the process but a guideline that is to be considered.

In addition, due to the nature of our experiments and the fact that we did not have access to real-life victims but mostly honeypots, our dataset only consists of the commands that the attacker executed. This selective data collection method inherently introduces a selection bias, as it does not capture commands from legitimate users typically present in real-world datasets [28]. Consequently, our model has a bias toward attacker commands, raising concerns about its effectiveness in real-world scenarios. To mitigate this bias and enhance the model's applicability to real-world scenarios, further research is needed. This could involve simulating more authentic datasets by incorporating legitimate user commands or utilizing actual user and attacker command datasets should such become publicly available.

We suggest exploring other influencing factors that dictate the behavior and tactics of threat actors. For example, one promising future research direction could be investigating the correlation between specific command sequences and the broader strategic goals of CTAs. It would be interesting to identify if a particular type of attack is more prevalent in the presence of a particular CTA. Additionally, it is beneficial to note that this study will contribute to coping with evolving threats; by analyzing the CTAs and their unique style, it is also possible to further research the emergent threats and their relations. Furthermore, it is important to gain more insights into how deep learning models determine the attribution of a CTA. As future work, we aim to explore the decision-making processes of these models, which could reveal the distinct patterns and behaviors they associate with specific CTAs. By interpreting the trained models, researchers and analysts could gain valuable insights that may not be immediately apparent from manual analysis alone.

9 Conclusion

In our study, we presented a novel approach to attributing CTA based on the sequence of commands they execute. This research was grounded in a unique intersection of NLP techniques and deep learning architectures, effectively distinguishing between various CTAs with high performance. We have developed a SCLC for CTA command sequence data, which effectively mitigates the issue of overfitting by transforming syntactic differences in the commands into a uniform format. This enhancement significantly improves the performance of our deep learning architecture.

Our hybrid deep learning architecture has proved to be a powerful solution for capturing local and global contextual information within the sequences of commands as it outperformed the notable pre-trained transformer models such as BERT, RoBERTa, SecureBERT, and DarkBERT. Hybrid architecture achieves an F1-score of 95.11% and an accuracy score of 95.13% on the high-count dataset, an F1-score of 93.60% and an accuracy score of 93.77% on the medium-count dataset, and an F1-score of 88.95% and accuracy score of 89.25% on the low-count dataset. These results indicate the potential of integrating CNNs and transformer mechanisms in complex NLP tasks, specifically within cybersecurity.

Our study resulted in several key findings that provide considerable insights into cybersecurity. First, we discovered that our approach could significantly reduce the workload of cybersecurity analysts by providing a systematic and automated means of attributing CTAs. Through the application of our method, the analysts are spared from manually tracing the origin of the threats. This allows them to concentrate their expertise and effort on more pressing or strategic tasks. Additionally, this automation not only enhances productivity but also reduces the risk of human error by providing an initial foothold for the threat actor attribution process.

With this knowledge, creating more effective countermeasures and defense strategies becomes much more feasible. Unique defense strategies can be developed based on the commands of the specific CTA. For instance, understanding a CTA's penchant for certain command sequences or attack vectors could guide the design of specific intrusion detection rules or firewall configurations. Similarly, knowing a CTA's pattern of behavior can also inform incident response plans, enabling swift and effective action in the unfortunate event of a security breach.

Furthermore, our research facilitates a deeper understanding of the modus operandi of specific CTAs. This nuanced understanding equips cybersecurity professionals with valuable knowledge to predict potential targets and anticipate the type of attack, thus enabling proactive rather than reactive defense strategies. Moreover, our work opens up multiple exciting avenues for future research within CTI and digital forensics. This includes investigating a deeper understanding of CTA behavior and the relationship between the sequence of commands and the strategic intent of the CTAs. These explorations could result in a more nuanced and thorough understanding of CTAs and their motivations.

Our research highlights the power and potential of NLP techniques and deep learning architectures in addressing complex digital forensic challenges. Our work represents a pivotal stepping stone towards an era where automated, intelligent systems play a central role in fortifying our cybersecurity infrastructure and simplifying digital forensics, as well as incident response challenges, by introducing a new method for CTA attribution. The pathway forward is filled with promising opportunities for innovative and groundbreaking research in cyber defense and threat intelligence research.

References

[1] Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. 2022. SecureBERT: A domain-specific language model for cybersecurity. In *Proceedings of the International Conference on Security and Privacy in Communication Systems*. Springer, 39–56.

[2] Aylin Caliskan, Fabian Yamaguchi, Edwin Dauber, Richard Harang, Konrad Rieck, Rachel Greenstadt, and Arvind Narayanan. 2015. When coding style survives compilation: De-anonymizing programmers from executable binaries. arXiv:1512.08546. Retrieved from https://doi.org/10.14722/ndss.2018.23304

[3] Cybersecurity and Infrastructure Security Agency (CISA). 2023. #StopRansomware: Royal ransomware. Retrieved from https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-061a

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. DOI: https://doi.org/10.18653/v1/N19-1423

[5] Juan Andres Guerrero-Saade. 2018. Draw me like one of your french APTs—Expanding our descriptive palette for cyber threat actors. In *Proceedings of the Virus Bulletin Conference*. 1–20.

[6] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. UNICORN: Runtime provenance-based detector for advanced persistent threats. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium (NDSS '20)*. DOI: https://doi.org/10.14722/ndss.2020.24046

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. arXiv:1512.03385. Retrieved from https://doi.org/10.48550/arXiv.1512.03385

[8] Health Sector Cybersecurity Coordination Center. 2022. HC3 Threat Profile: Evil Corp. U.S. Department of Health and Human Services. Retrieved from https://www.hhs.gov/sites/default/files/evil-corp-threat-profile.pdf

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

[10] Ehtsham Irshad and Abdul Basit Siddiqui. 2023. Cyber threat attribution using unstructured reports in cyber threat intelligence. *Egyptian Informatics Journal* 24, 1 (2023), 43–59.

[11] Youngjin Jin, Eugene Jang, Jian Cui, Jin-Woo Chung, Yongjae Lee, and Seungwon Shin. 2023. DarkBERT: A language model for the dark side of the internet. arXiv:2305.08596. Retrieved from https://doi.org/10.48550/arXiv.2305.08596

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[13] Yann LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1, 4 (1989), 541–551. DOI: https://doi.org/10.1162/neco.1989.1.4.541

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692. Retrieved from https://doi.org/10.48550/arXiv.1907.11692

[15] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. arXiv:1711.05101. Retrieved from https://doi.org/10.48550/arXiv.1711.05101

[16] Sadegh M. Milajerdi, Rigel Gjomemo, Birhanu Eshete, R. Sekar, and V. N. Venkatakrishnan. 2019. HOLMES: Real-time apt detection through correlation of suspicious information flows. arXiv:1810.01594. Retrieved from https://doi.org/10.48550/arXiv.1810.01594

[17] Umara Noor, Zahid Anwar, Tehmina Amjad, and Kim-Kwang Raymond Choo. 2019. A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise. *Future Generation Computer Systems* 96 (2019), 227–242.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. arXiv:1912.01703. Retrieved from https://doi.org/10.48550/arXiv.1912.01703

[19] Lior Perry, Bracha Shapira, and Rami Puzis. 2019. NO-DOUBT: Attack attribution based on threat intelligence reports. In *Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 80–85. DOI: https://doi.org/10.1109/ISI.2019.8823152

[20] Avi Pfeffer, C. Call, John Chamberlain, L. Kellogg, Jacob Ouellette, T. Patten, Greg Zacharias, Arun Lakhotia, Suresh Golconda, J. Bay, Robert Hall, and Daniel Scofield. 2012. Malware analysis and attribution using genetic information. In *Proceedings of the 2012 7th International Conference on Malicious and Unwanted Software*. 39–45. DOI: https://doi.org/10.1109/MALWARE.2012.6461006.

[21] PRODAFT. 2022. [ws] Wizard Spider Group In-Depth Analysis. Retrieved from https://www.prodaft.com/resource/detail/ws-wizard-spider-group-depth-analysis

[22] PRODAFT. 2021. Silverfish: Global Cyber Espionage Campaign Case Report. Retrieved from https://www.prodaft.com/resource/detail/silverfish-global-cyber-espionage-campaign-case-report

[23] Nathan Rosenblum, Xiaojin Zhu, and Barton P. Miller. 2011. Who wrote this code? Identifying the authors of program binaries. In *Proceedings of the 16th European Conference on Research in Computer Security (ESORICS '11)*. Vijay Atluri and Claudia Diaz (Eds.), Springer, Berlin, 172–189.

[24] S. Naveen, Rami Puzis, and Kumaresan Angappan. 2020. Deep learning for threat actor attribution from threat reports. In *Proceedings of the 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. 1–6. DOI: https://doi.org/10.1109/ICCCSP49186.2020.9315219.

[25] Md Sahrom, S. Rahayu, Aswami Ariffin, and Y. Robiah. 2018. Cyber threat intelligence – Issue and challenges. *Indonesian Journal of Electrical Engineering and Computer Science* 10, 1 (2018), 371–379. DOI: https://doi.org/10.11591/ijeecs.v10.i1.pp371-379

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[27] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. 2015. Efficient object localization using convolutional networks. arXiv:1411.4280. Retrieved from https://doi.org/10.48550/arXiv.1411.4280

[28] Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*. IEEE, 1521–1528.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

Received 30 December 2023; revised 11 May 2024; accepted 27 June 2024

Automated ATT&CK Technique Chaining

GEIR SKJØTSKIFT, MARTIN EIAN, and SIRI BROMANDER, mnemonic AS, Oslo, Norway

Incident response teams need to determine what happened before and after an observation of adversary behavior in order to effectively respond to incidents. The MITRE ATT&CK knowledge base provides useful information about adversary behaviors but provides no guidance on what most likely happened before and after an observed behavior. We have developed methods and open source tools to help incident responders answer the questions "What did most likely happen prior to this observation?" and "What are the adversary's most likely next steps given this observation?" To be able to answer these questions, we combine semantic modeling of subject matter expert knowledge with data-driven methods trained on data from computer security incidents.

CCS Concepts: • **Mathematics of computing** → **Markov-chain Monte Carlo methods**; • **Security and privacy** → **Formal security models**; • **Applied computing** → **Investigation techniques**;

Additional Key Words and Phrases: ATT&CK, Incident response, Technique chains

ACM Reference format:
Geir Skjøtskift, Martin Eian, and Siri Bromander. 2025. Automated ATT&CK Technique Chaining. *Digit. Threat. Res. Pract.* 6, 1, Article 3 (February 2025), 11 pages.
https://doi.org/10.1145/3696013

1 Introduction

Understanding adversary behavior is important for effective incident response. Incident response teams need to determine what happened before and after an observation in order to assess the scope of a breach and respond to an incident. A key resource for understanding adversary behavior is MITRE's **Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)** [12], a taxonomy and knowledge base of adversary behaviors. Adversaries use several different techniques in computer security incidents, and these techniques are related to each other. One shortcoming of the ATT&CK framework is that it provides no guidance on technique sequences or dependencies. Given an observation of adversary behavior, it is therefore difficult to do automated reasoning about what happened before and about what will most likely happen next.

Our organization has, over the last two decades, responded to hundreds of incidents across all major sectors. We have observed that machine-readable models of the victim infrastructure are rarely, if ever, available to incident responders. Methods that rely on such models are thus not applicable to the vast majority of incidents.

The research described in Section 2 was carried out in the SOCCRATES project, funded by the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No. 833481. The research described in Section 3 was carried out in the CyberHunt project, funded by the IKTPLUSS program of the Research Council of Norway under Grant Agreement No. 303585.

Authors' Contact Information: Geir Skjøtskift, mnemonic AS, Oslo, Norway; e-mail: geir@mnemonic.no; Martin Eian (corresponding author), mnemonic AS, Oslo, Norway; e-mail: meian@mnemonic.no; Siri Bromander, mnemonic AS, Oslo, Norway; e-mail: siri@mnemonic.no.

ACM 2576-5337/2025/2-ART3
https://doi.org/10.1145/3696013

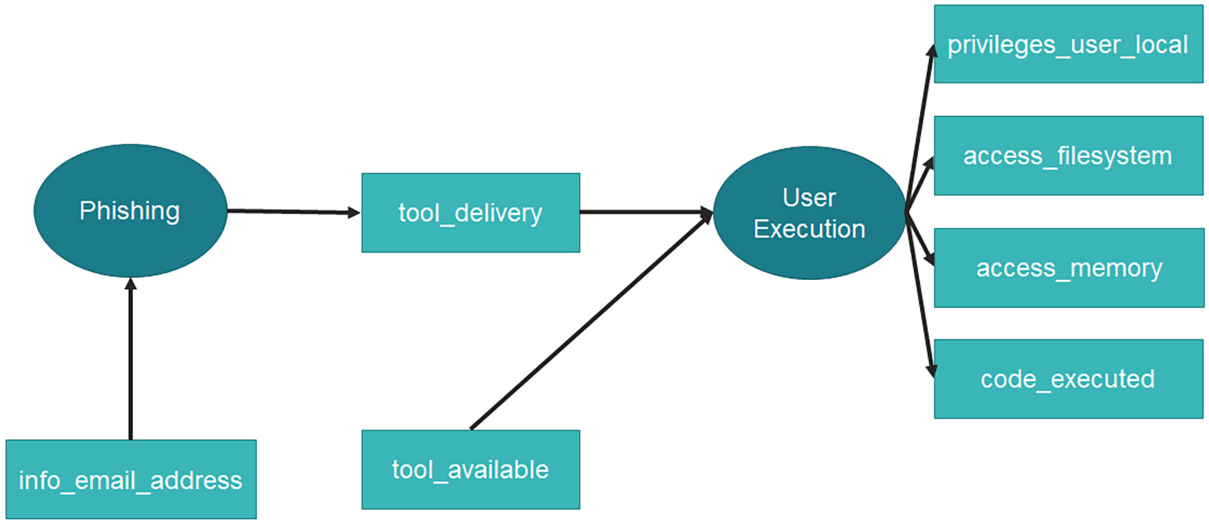


Fig. 1. Phishing requires information about a valid target e-mail address (info_email_address), and it provides the ability to deliver an arbitrary payload to a target system (tool_delivery). User Execution requires access to malicious software (tool_available) and tool_delivery. Once its requirements are met, User Execution provides local user privileges, filesystem access, memory access, and code execution.

In this work, we present our methods and our open-source tools for generating attack stages from lists of techniques, as well as generating the most likely technique chain given an observed technique. The tools can help incident responders answer the questions "What did most likely happen prior to this observation?" and "What are the adversary's most likely next steps given this observation?" Furthermore, the tools can automate parts of the adversary emulation process for red teams. Our key contributions are:

- —A semantic model of technique dependencies
- —A vocabulary of abilities required by and provided by the techniques in ATT&CK
- —Mapping the abilities of all techniques and sub-techniques in ATT&CK
- —A data-driven model for generating the most likely chains of techniques
- —Open source tools that implement the semantic and data-driven models

The rest of this article is structured as follows: Section 2 presents our semantic model of technique dependencies and results, Section 3 presents our data-driven model and results, and Section 4 presents related work. Finally, Section 5 contains the discussion of our results, and Section 6 contains our conclusions.

2 A Semantic Model of Technique Dependencies

A technique *requires* a set of abilities in order to be executed, e.g., information about network services, access to credentials, or access to exploit code. A technique also *provides* a set of abilities, which in turn might be required by other techniques. A simple example of this concept using the techniques of Phishing (T1566) and User Execution (T1204) is illustrated in Figure 1.

We apply promise theory [4] to create a formal model of technique dependencies based on subject matter expert knowledge. Promise theory supports modeling of implicit coordination between autonomous agents, which we use to automatically generate a set of attack stages from a set of techniques. The use of promise theory to model techniques as autonomous agents lets us analyze an ATT&CK technique and assign abilities to it without explicitly considering its dependencies on other techniques. This enables a team of analysts to work in parallel,

Algorithm 1: Generating Attack Stages from a List of Techniques and an Objective

```
Data: $T \subseteq \mathcal{T}$, $o \in \mathcal{O}$, $R$, $P$
Result: $S$
$i \leftarrow 1$;
$Q \leftarrow \varnothing$;
$S \leftarrow \varnothing$;
$T' \leftarrow \varnothing$;
while $\exists t \in T : \{a : (t, a) \in R\} \subseteq Q$ do
    for $t \in T$ do
        if $\{a : (t, a) \in R\} \subseteq Q$ then
            $Q \leftarrow Q \cup \{x : (t, x) \in P\}$;
            $T' \leftarrow T' \cup \{t\}$
        end
    end
    $S \leftarrow S \cup \{(i, T')\}$;
    $i \leftarrow i + 1$;
    $T \leftarrow T \setminus T'$;
    $T' \leftarrow \varnothing$;
end
```

where each analyst is assigned a set of techniques to review. When new techniques are added to ATT&CK, abilities can be assigned to them without reviewing the entire library of techniques. Our application of promise theory is thus both scalable and maintainable.

We define the set $\mathcal{T}$ of techniques and sub-techniques in ATT&CK as autonomous agents. Henceforth, we will use the term "technique" to refer to both techniques and sub-techniques. In order to apply promise theory, we also need a vocabulary of promises. We define $\mathcal{A}$ as the set of abilities required by or provided by techniques, and $\mathcal{O}$ as the set of adversary objectives. The set of abilities and objectives that we have developed is available in Appendix A and on GitHub [18].

We use a team of eight subject matter experts with extensive experience in the fields of detection engineering, incident response, threat intelligence, and threat hunting to review the techniques. For every technique in ATT&CK, we let one pair of experts review the technique and procedure descriptions and then assign the required abilities and provided abilities and/or objectives. Then, the other pairs of experts assess their results and suggest improvements. We record the abilities as the relations R (requires) and P (provides):

$$R \subset \mathcal{T} \times \mathcal{A}, \tag{1}$$

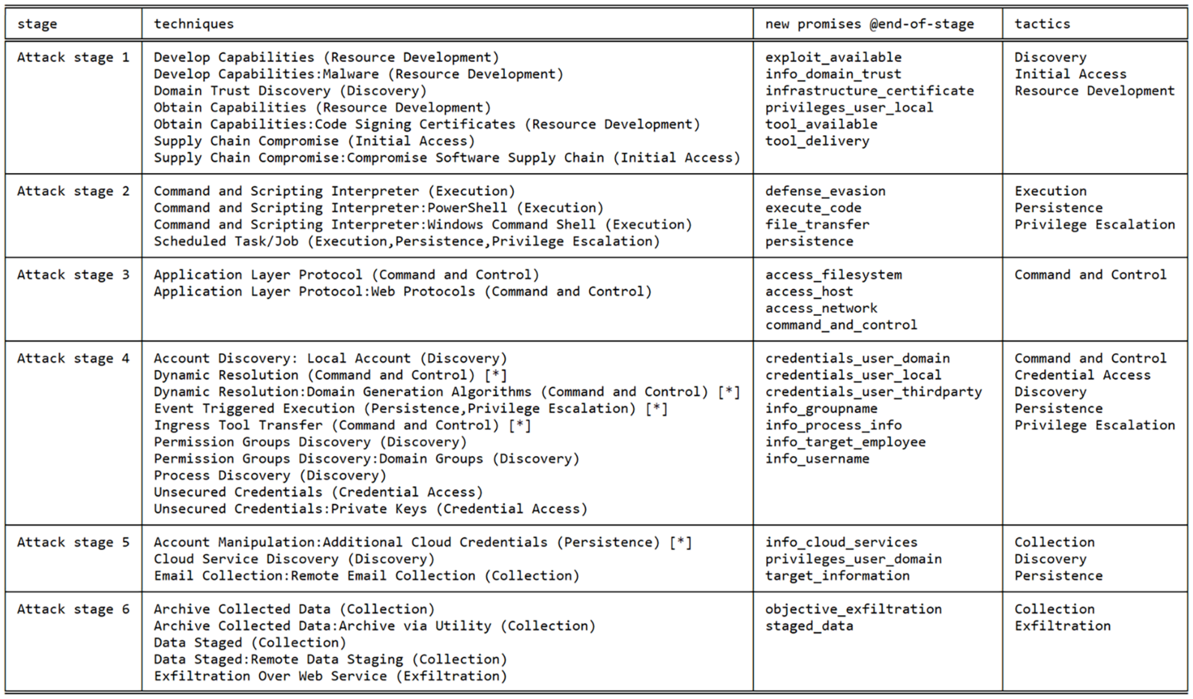
$$P \subset \mathcal{T} \times (\mathcal{A} \cup \mathcal{O}). \tag{2}$$

The relations R and P are also available on GitHub [17]. Finally, we define the promises made by each $t \in \mathcal{T}$:

$$t \xrightarrow{\{x:(t,x)\in P\}\,|\,\{a:(t,a)\in R\}} * \tag{3}$$

$$t \xrightarrow{U(\{a:(t,a)\in R\})} * \tag{4}$$

Equation (3) is a conditional promise, while Equation (4) is a promise to use the abilities required by t if received. These two equations define the behavior of the techniques $t \in \mathcal{T}$. We use the notation defined in [4]: $a|b$ is a conditional promise, where the agent promises to provide a if it receives b. The target $*$ represents any agent or all agents, and the notation $U(b)$ is a promise by the agent to use b if it receives b.



stage	techniques	new promises @end-of-stage	tactics
Attack stage 1	Develop Capabilities (Resource Development) Develop Capabilities:Malware (Resource Development) Domain Trust Discovery (Discovery) Obtain Capabilities (Resource Development) Obtain Capabilities:Code Signing Certificates (Resource Development) Supply Chain Compromise (Initial Access) Supply Chain Compromise:Compromise Software Supply Chain (Initial Access)	exploit_available info_domain_trust infrastructure_certificate privileges_user_local tool_available tool_delivery	Discovery Initial Access Resource Development
Attack stage 2	Command and Scripting Interpreter (Execution) Command and Scripting Interpreter:PowerShell (Execution) Command and Scripting Interpreter:Windows Command Shell (Execution) Scheduled Task/Job (Execution,Persistence,Privilege Escalation)	defense_evasion execute_code file_transfer persistence	Execution Persistence Privilege Escalation
Attack stage 3	Application Layer Protocol (Command and Control) Application Layer Protocol:Web Protocols (Command and Control)	access_filesystem access_host access_network command_and_control	Command and Control
Attack stage 4	Account Discovery: Local Account (Discovery) Dynamic Resolution (Command and Control) [*] Dynamic Resolution:Domain Generation Algorithms (Command and Control) [*] Event Triggered Execution (Persistence,Privilege Escalation) [*] Ingress Tool Transfer (Command and Control) [*] Permission Groups Discovery (Discovery) Permission Groups Discovery:Domain Groups (Discovery) Process Discovery (Discovery) Unsecured Credentials (Credential Access) Unsecured Credentials:Private Keys (Credential Access)	credentials_user_domain credentials_user_local credentials_user_thirdparty info_groupname info_process_info info_target_employee info_username	Command and Control Credential Access Discovery Persistence Privilege Escalation
Attack stage 5	Account Manipulation:Additional Cloud Credentials (Persistence) [*] Cloud Service Discovery (Discovery) Email Collection:Remote Email Collection (Collection)	info_cloud_services privileges_user_domain target_information	Collection Discovery Persistence
Attack stage 6	Archive Collected Data (Collection) Archive Collected Data:Archive via Utility (Collection) Data Staged (Collection) Data Staged:Remote Data Staging (Collection) Exfiltration Over Web Service (Exfiltration)	objective_exfiltration staged_data	Collection Exfiltration

Fig. 2. An example tool execution, using an ATT&CK Navigator Layer from the SolarWinds incident. There are six attack stages, and at stage six, the adversary achieves objective_exfiltration. The first column shows the attack stages, the second shows techniques executed, the third shows the new abilities and objectives added, and the fourth shows the tactics associated with the techniques in each stage.

Having defined the set of autonomous agents and their behavior, developed the vocabularies of abilities and objectives, and defined the relations to specific techniques, we implement an open source tool to generate attack stages from a list of techniques. The tool is available on GitHub [16]. Informally, the tool takes a set of techniques, an objective, and the "requires" and "provides" relations as input, and returns a list of attack stages with the associated techniques as output. For each stage, the tool executes every technique where all "requires" abilities are present in a set Q, then adds the "provides" abilities and/or objectives from those techniques to Q. The set Q is thus the communication channel where the techniques publish their promises. Note that since Q is initially empty, only techniques with no requirements can be executed in the first stage. Algorithm 1 shows the pseudo-code for the core functionality of the tool.

We tested the tool on data from public incident reports and on the techniques associated with the different Adversary Groups in ATT&CK. Figure 2 shows the output from an example tool execution. Initial results showed that a significant number of techniques were missing from the Adversary Groups in the ATT&CK knowledge base, in particular techniques from the Reconnaissance and Resource Development tactics. Based on our tests, we contributed missing techniques back to ATT&CK [11]. We also found the tool to be useful when planning red team exercises since it can automate parts of the time-consuming process of developing an operational flow for an adversary emulation plan [8]. You give the tool a list of techniques, and it will generate the attack stages for you.

Note that our semantic model and tool only provide possibilities; they do not provide any probabilities. Each attack stage gives you a list of *possible* techniques, and these stages can result in a large number of possible attack chains. Thus, the semantic model is not able to answer the questions posed in the Introduction. In order to answer

Algorithm 2: Counting How Many Times a Technique Has Provided an Ability

```
Data: $\mathcal{I}$, $R$, $P$
Result: $\mathcal{S}$, counter
$\mathcal{S} \leftarrow \emptyset$;
for $I \in \mathcal{I}$ do
    for $t \in I$ do
        for $y \in \{x : (t, x) \in P\}$ do
            if $\exists t' \in I : (t', y) \in R$ then
                if $(y, t) \in \mathcal{S}$ then
                    increment counter for the tuple $(y, t) \in \mathcal{S}$ by 1;
                else
                    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(y, t)\}$;
                    set counter for the tuple $(y, t) \in \mathcal{S}$ equal to 1;
                end
            end
        end
    end
end
```

the questions, we combine the semantic model with a data-driven model trained on observations of techniques used in computer security incidents.

3 A Data-Driven Model for Technique Chains

Our data-driven model is based on technique *co-occurrences*. A technique co-occurrence is a set of techniques that have been observed in the same incident. Formally, for the set $\mathcal{I}$ of incidents:

$$\forall I \in \mathcal{I} : I \subseteq \mathcal{T}. \tag{5}$$

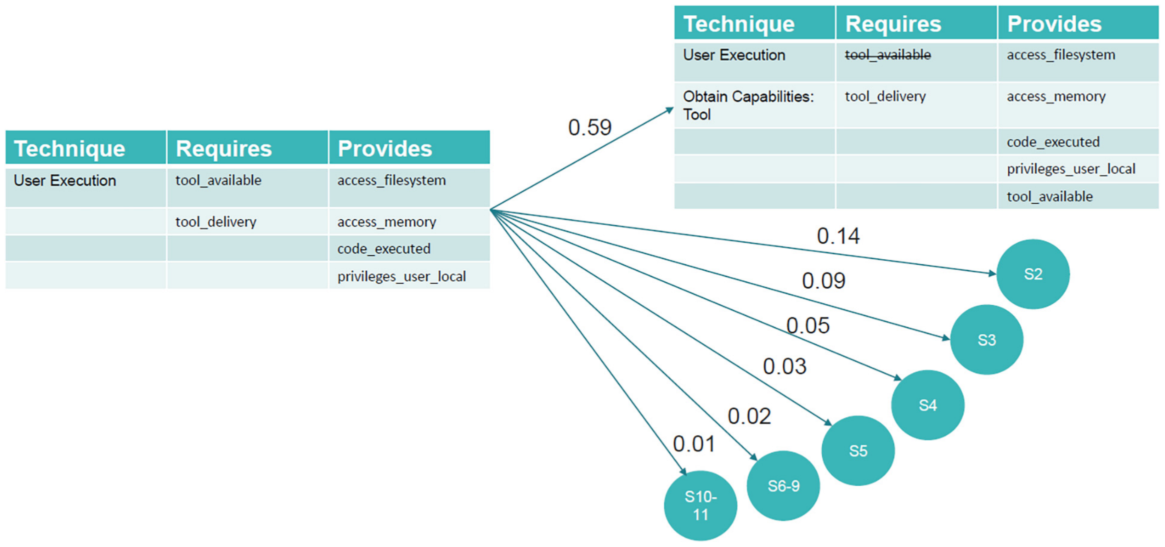
We collect incident reports with ATT&CK techniques from several different data sources: MITRE ATT&CK Campaigns [13], MITRE ATT&CK Groups [14], MITRE Engenuity's Adversary Emulation Plan Library [8], Palo Alto Unit 42 Adversary Playbooks [1], The DFIR Report [21], ATT&CK Navigator Layers from published incident reports, and data from incidents that we have handled ourselves. A single incident report from a single data source represents an incident $I \in \mathcal{I}$. We then use the semantic model described in Section 2 to count the number of times a technique has provided an ability $a \in \mathcal{A}$ that was required by another technique observed in the same incident I. Algorithm 2 shows the pseudo-code for counting the number of times a technique has provided an ability.

After executing the tool and implementing the algorithm, we get a JavaScript Object Notation file that for each ability $a \in \mathcal{A}$ lists the techniques that have provided the ability and a count of how many times each technique has provided the ability. Figure 3 shows the technique counts for the ability "tool_available."

We then proceed to use the technique counts to construct a Markov chain, where we define the states as $(T \subseteq \mathcal{T}, A \subseteq \mathcal{A}, X \subseteq (\mathcal{A} \cup \mathcal{O}))$. Each state contains a set of techniques T, the set A of abilities required by the techniques in T that have not yet been provided, and the set X of abilities and/or objectives provided by the techniques in T. Since we model techniques as autonomous agents from promise theory, where each technique acts independently, the Markov property holds. Given an observation of a technique, this Markov chain can be used to find the most likely attack chain leading to that observation, thus answering the question "What did most likely happen prior to this observation?" Figure 4 shows a simple example of Markov chain states, using the

```
"tool_available": {
  "T1588.002": 59,
  "T1587.001": 14,
  "T1588.001": 9,
  "T1588.003": 5,
  "T1587": 3,
  "T1587.002": 2,
  "T1587.003": 2,
  "T1588": 2,
  "T1588.004": 2,
  "T1588.005": 1,
  "T1588.006": 1
}
```

Fig. 3. The number of times that techniques have provided the ability tool_available. In total, we observed 100 instances of a technique providing the ability tool_available to another technique observed in the same incident. The most frequently observed technique is Obtain Capabilities: Tool (T1588.002) with 59 occurrences. We thus compute that if a technique requires tool_available, then that ability was provided by T1588.002 with probability p = 0.59.



Technique	Requires	Provides
User Execution	tool_available	access_filesystem
	tool_delivery	access_memory
		code_executed
		privileges_user_local

Technique	Requires	Provides
User Execution	~~tool_available~~	access_filesystem
Obtain Capabilities: Tool	tool_delivery	access_memory
		code_executed
		privileges_user_local
		tool_available

Fig. 4. A simple Markov chain example. The initial state to the left shows the observed technique User Execution (T1204), along with the abilities required by the technique and the abilities provided by the technique. The edges show transitions to states where the tool_available ability is provided by another technique, as well as the transition probabilities computed from the training data. In this example, tool_available will be provided by the technique Obtain Capabilities: Tool (T1588.002) with probability p = 0.59.

transition probabilities computed from the data in Figure 3 and starting from the initial state where the technique User Execution (T1204) is observed.

In order to find the most likely path to the observed technique, we have to follow every available transition. The state space therefore increases exponentially for each ability provided. This makes construction of the complete Markov chain infeasible, so we use Markov chain Monte Carlo simulations to find the most likely path. The starting

```
69.77% (n=69769)
╔═══════╦═════════════════════════════╦═══════════════════════════╦════════════════╗
║ stage ║ techniques                  ║ new promises @end-of-stage║ tactics        ║
╠═══════╬═════════════════════════════╬═══════════════════════════╬════════════════╣
║     1 ║ Phishing (Initial Access)   ║ credentials_user_domain   ║ Initial Access ║
║       ║                             ║ credentials_user_local    ║                ║
║       ║                             ║ tool_delivery             ║                ║
╟───────╫─────────────────────────────╫───────────────────────────╫────────────────╢
║     2 ║ User Execution (Execution)  ║ access_filesystem         ║ Execution      ║
║       ║                             ║ access_memory             ║                ║
║       ║                             ║ code_executed             ║                ║
║       ║                             ║ privileges_user_local     ║                ║
╚═══════╩═════════════════════════════╩═══════════════════════════╩════════════════╝
```

Fig. 5. Markov chain Monte Carlo simulation results for the observed technique User Execution (T1204). The most likely attack chain given this observation is Phishing followed by User Execution (p = 0.6977). Note that this simulation was run with pre-seeding of Reconnaissance and Resource Development techniques, since these techniques are rarely observable by incident responders.

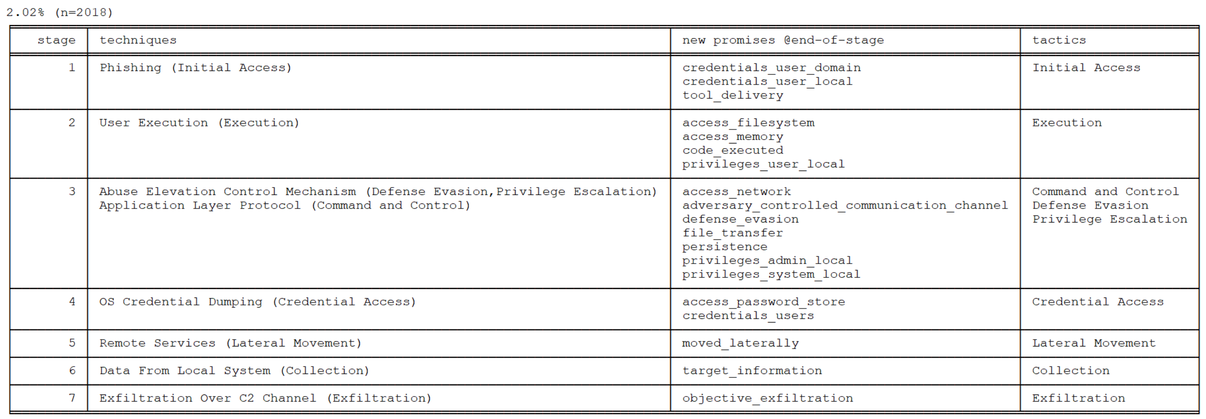
state is one or more observed techniques, along with the abilities required and provided. For each iteration, we pick the first ability $a \in A$ required by the current state and then pick a transition at random according to the transition probabilities. We continue doing this until no more transitions are possible or until $A = \emptyset$. Finally, we use the tool described in Section 2 to validate the resulting attack chain. If the chain is not valid, i.e., possible to execute using the semantic model implementation, then we discard it and start over. If the chain is valid, then we increment the count of how often this unique chain has been observed. We then repeat this until we have 100.000 valid attack chains and then compute the frequency of each unique chain observed. We have developed open source tools that implement all of the functionality described in this Section and made them available on GitHub [19]. The tools also support the identification of choke points, i.e., the techniques most frequently observed in the attack chains. Figure 5 shows an example simulation output of the most likely attack chain given the observation of the technique User Execution (T1204).

The tools can also help incident responders answer the question "What are the adversary's most likely next steps given this observation?" As an example, given the observation of User Execution, we can also provide the tool with a candidate target technique, e.g., Exfiltration over C2 Channel. Figure 6 shows an example simulation output of this scenario. In this case, the probability of the most likely attack chain is very low due to insufficient training data and the length of the attack chain.

4 Related Work

The literature on the topic of ATT&CK technique chains is sparse. Al-Shaer et al. used clustering methods to try to learn associations between techniques [2]. Their approach differs significantly from ours in that they did not capture or use subject matter expert knowledge. Furthermore, they used a much smaller set of training data than our research, and they categorized Software in MITRE ATT&CK as "attacks," which might give misleading results. The Software entries in ATT&CK describe the total set of techniques implemented by a malware or tool family, they do not represent techniques observed in computer security incidents.

Kim et al. proposed an attack prediction model based on a Bayesian network [10]. Their goals include the prediction of the most likely next ATT&CK technique given observations of executed techniques in an incident. Their method is purely data-driven, as opposed to our combination of semantic and data-driven methods, and their paper does not address bias in the reports used to train their model. They provide a quantitative assessment of their model's performance, but the datasets used do not provide the ground truth needed to make such an assessment. To the best of our knowledge, no such ground truth is available.



2.02% (n=2018)

stage	techniques	new promises @end-of-stage	tactics
1	Phishing (Initial Access)	credentials_user_domain credentials_user_local tool_delivery	Initial Access
2	User Execution (Execution)	access_filesystem access_memory code_executed privileges_user_local	Execution
3	Abuse Elevation Control Mechanism (Defense Evasion,Privilege Escalation) Application Layer Protocol (Command and Control)	access_network adversary_controlled_communication_channel defense_evasion file_transfer persistence privileges_admin_local privileges_system_local	Command and Control Defense Evasion Privilege Escalation
4	OS Credential Dumping (Credential Access)	access_password_store credentials_users	Credential Access
5	Remote Services (Lateral Movement)	moved_laterally	Lateral Movement
6	Data From Local System (Collection)	target_information	Collection
7	Exfiltration Over C2 Channel (Exfiltration)	objective_exfiltration	Exfiltration

Fig. 6. Markov chain Monte Carlo simulation results for the observed techniques User Execution (T1204) and the candidate target technique Exfiltration over C2 Channel (T1041). The most likely attack chain for this scenario has the probability p = 0.0202. Note that this simulation was run with pre-seeding of Reconnaissance and Resource Development techniques since these techniques are rarely observable by incident responders.

Andy Appelbaum presented a method for semantic modeling of technique dependencies at the 31st Annual FIRST Conference in 2019 [3]. His research only covered a small subset of the techniques in MITRE ATT&CK, but this presentation inspired our work on semantic modeling of technique dependencies.

MITRE Caldera [15] implements a semantic model of technique dependencies. This implementation is specific to the tool and cannot be generalized to cover all of the techniques in MITRE ATT&CK, while our semantic model covers all of the techniques in ATT&CK version 12.

MITRE Engenuity's Attack Flow provides a language to describe flows of ATT&CK techniques [6]. Attack Flow does not, however, provide the semantics to generate such flows. We have had discussions with the developers of Attack Flow on how to use our tool to generate a library of example flows for Attack Flow.

Other related literature describes the use of attack trees and attack graphs to determine the probability of different attack paths [5, 9, 20]. These approaches depend on a machine-readable model of the victim infrastructure and security controls, which is rarely available to incident responders.

5 Discussion

We have extensively tested the tools implementing both the semantic model and the data-driven model, and we have spent a large amount of effort to verify that the results are not erroneous or misleading. However, it is important to be aware of potential weaknesses and sources of errors.

For the semantic model, the main sources of errors are cognitive bias and human errors. Mapping abilities to every technique in ATT&CK required extensive effort by human subject matter experts. We addressed this by letting experts work in pairs and by letting other pairs of experts review the mappings. We also tested the tool iteratively to validate that the results made sense to a human expert. The iteration loop of developing the vocabulary, mapping abilities to techniques, and tool testing proceeded until the resulting attack stages corresponded to what we would expect to observe in real incidents. Furthermore, the tool and all the mappings are published on GitHub under an open source license, so any interested party can reproduce our results and suggest improvements.

The data-driven model is also subject to bias. Detection of techniques depends on visibility, detection coverage, and mapping of detection analytics to ATT&CK. Furthermore, incident reports written by human analysts are subject to the same biases as described in the paragraph above. As a result, techniques that are easier to detect and classify receive a greater weight in the model. In order to address these biases, we need to improve visibility, detection coverage, and classification of attacks.

Another major issue with the data-driven model is the lack of training data. We decided to use Groups in ATT&CK as a data source, which is not ideal. Groups represent all techniques that have been observed used by a specific threat actor, in some cases over a time period of more than a decade. Ideally, the technique co-occurrence data should be separated per incident, but this is not possible with the Group data from ATT&CK. A newer category of data in ATT&CK is Campaigns, which is much closer to what we need. Over time, this category will grow and provide more training data. Currently, if an attack chain is long, e.g., when the initial observation is a technique from the Exfiltration tactic, then the most likely attack chain has a very low probability. This is in part due to the lack of training data. We have implemented different aggregation strategies, such as aggregating sub-techniques to techniques and aggregating equivalent techniques, to alleviate this issue. However, these strategies are no substitute for more training data. We currently reach out to various organizations to try to convince them to share their data on technique co-occurrences. Furthermore, we have joined the Sightings Ecosystem [7] as a contributor in order to get access to more technique co-occurrence data. The lack of training data also means that using more advanced machine learning methods is not currently feasible since such methods require much larger amounts of training data than our approach.

As mentioned in the Introduction, a machine-readable model of the victim infrastructure is rarely available to incident responders. The victim infrastructure and security controls could have a substantial influence on an adversary's behavior, and thus on the probabilities of the attack chains. This influence is not captured by our model.

One of the major strengths of our semantic model is its scalability and maintainability. Since we model techniques as independent, autonomous agents, an analyst can review a new technique added to ATT&CK without having to consider its impact on other techniques. New techniques are added with every new version of ATT&CK, so a model without this independence assumption would require an excessive amount of effort to maintain.

6 Conclusions

We have presented our methods and open source tools to help incident responders answer the questions "What did most likely happen prior to this observation?" and "What are the adversary's most likely next steps given this observation?" We use a combination of semantic modeling of subject matter expert knowledge and data-driven methods to achieve this goal. The tools we have developed are readily available as open source software on GitHub.

Acknowledgments

We would like to thank Fredrik Borg, Fredrik Morten Sætran, Mona Elisabeth Østvang, Philip Christian Scheel, and Tom Spangebu for their contributions to the review of techniques and procedures in MITRE ATT&CK and other research described in this article. We would also like to thank Peter Flydal for his valuable feedback on notation for the equations and algorithms.

References

[1] Palo Alto Unit 42. 2023. Unit 42 Adversary Playbooks. Retrieved September 24, 2024 from https://github.com/pan-unit42/playbook_viewer/tree/master/playbook_json
[2] Rawan Al-Shaer, Jonathan M. Spring, and Eliana Christou. 2020. Learning the associations of MITRE ATT&CK adversarial techniques. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, 1–9. DOI: https://doi.org/10.1109/CNS48642.2020.9162207
[3] Andy Appelbaum. 2019. Finding Dependencies Between Adversary Techniques. Retrieved September 24, 2024 from https://www.first.org/resources/papers/conf2019/1100-Applebaum.pdf
[4] Jan A. Bergstra and Mark Burgess. 2014. *Promise Theory: Principles and Applications* (2nd ed.). Independently Published.

[5] Christian Ellerhold, Johann Schnagl, and Thomas Schreck. 2023. Enterprise cyber threat modeling and simulation of loss events for cyber risk quantification. In *Proceedings of the Cloud Computing Security Workshop*, 17–29.

[6] MITRE Engenuity Center for Threat-Informed Defense. 2022. Attack Flow. Retrieved September 24, 2024 from https://mitre-engenuity.org/cybersecurity/center-for-threat-informed-defense/our-work/attack-flow/

[7] MITRE Engenuity Center for Threat-Informed Defense. 2022. Sightings Ecosystem. Retrieved September 24, 2024 from https://github.com/center-for-threat-informed-defense/sightings_ecosystem

[8] MITRE Engenuity Center for Threat-Informed Defense. 2023. Adversary Emulation Library. Retrieved September 24, 2024 from https://github.com/center-for-threat-informed-defense/adversary_emulation_library

[9] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. 2018. A meta language for threat modeling and attack simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 1–8.

[10] Youngjoon Kim, Insup Lee, Hyuk Kwon, Kyeongsik Lee, and Jiwon Yoon. 2023. *BAN: Predicting APT Attack Based on Bayesian Network With MITRE ATT&CK Framework*, Vol. 11. IEEE, 91949–91968. DOI: https://doi.org/10.1109/ACCESS.2023.3306593

[11] MITRE. 2021. Obtain Capabilities: Tool. Retrieved September 24, 2024 from https://attack.mitre.org/techniques/T1588/002/

[12] MITRE. 2023. MITRE ATT&CK. Retrieved September 24, 2024 from https://attack.mitre.org/

[13] MITRE. 2023. MITRE ATT&CK Campaigns. Retrieved September 24, 2024 from https://attack.mitre.org/campaigns/

[14] MITRE. 2023. MITRE ATT&CK Groups. Retrieved September 24, 2024 from https://attack.mitre.org/groups/

[15] MITRE. 2023. MITRE Caldera. Retrieved September 24, 2024 from https://caldera.mitre.org/

[16] mnemonic. 2021. Adversary Emulation Planner. Retrieved September 24, 2024 from https://github.com/mnemonic-no/provreq

[17] mnemonic. 2021. Techniques and Promises. Retrieved September 24, 2024 from https://github.com/mnemonic-no/aep/blob/master/data/agent_promises.json

[18] mnemonic. 2021. Vocabulary of Promises. Retrieved September 24, 2024 from https://github.com/mnemonic-no/aep/blob/master/data/promise_descriptions.csv

[19] mnemonic. 2023. Markov chain Monte Carlo Tool. Retrieved September 24, 2024 from https://github.com/mnemonic-no/provreq-mcmc

[20] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. 2005. MulVAL: A logic-based network security analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium*, Vol. 8, 113–128.

[21] The DFIR Report. 2023. The DFIR Report. Retrieved September 24, 2024 from https://thedfirreport.com/

Appendix

A Set of Abilities and Objectives

This appendix contains the set of abilities and objectives described in Section 2. Entries with the prefix “objective_” are adversary objectives, the rest are abilities.

```
access_filesystem
access_memory
access_network
access_network_infrastructure
access_network_intercept
access_password_store
access_physical
adversary_controlled_communication_channel
code_executed
credentials_2fa_token
credentials_admin_domain
credentials_admin_local
credentials_user_domain
credentials_user_local
credentials_user_thirdparty
credentials_users
defense_evasion
exploit_available
exploit_delivery
fast_flux
file_transfer
```

```
info_cloud_hosts
info_cloud_services
info_domain_trust
info_email_address
info_groupname
info_network_config
info_network_hosts
info_network_services
info_network_shares
info_network_trust
info_password_policy
info_peripheral_devices
info_process_info
info_target_information
info_service_info
info_system_time
info_target_employee
info_username
info_vulnerability
infrastructure_botnet
infrastructure_certificate
infrastructure_domain
infrastructure_server
infrastructure_trusted_email_account
infrastructure_trusted_social_media
moved_laterally
objective_denial_of_service
objective_destruction
objective_exfiltration
objective_extortion
objective_integrity
objective_resources_computational
persistence
privileges_admin_domain
privileges_admin_local
privileges_system_local
privileges_user_domain
privileges_user_local
privileges_user_thirdparty
privileges_users
staged_data
target_information
tool_available
tool_delivery
waterhole
```

Received 21 December 2023; revised 10 June 2024; accepted 19 August 2024

FedNIDS: A Federated Learning Framework for Packet-Based Network Intrusion Detection System

QUOC H. NGUYEN and SOUMYADEEP HORE, University of South Florida, Tampa, Florida, USA
ANKIT SHAH, Indiana University, Bloomington, Indiana, USA
TRUNG LE, University of South Florida, Tampa, Florida, USA
NATHANIEL D. BASTIAN, United States Military Academy, West Point, New York, USA

Network intrusion detection systems (NIDS) play a critical role in discerning between benign and malicious network traffic. Deep neural networks (DNNs), anchored on large and diverse datasets, exhibit promise in enhancing the detection accuracy of NIDS by capturing intricate network traffic patterns. However, safeguarding distributed computer networks against emerging cyber threats is increasingly challenging. Despite the abundance of diverse network data, decentralization persists due to data privacy and security concerns. This confers an asymmetric advantage to adversaries, as distributed networks face the formidable task of securely and efficiently sharing non-independently and identically distributed data to counter cyber-attacks. To address this, we propose the federated NIDS (*FedNIDS*), a novel two-stage framework that combines the power of federated learning and DNNs. It aims to enhance the detection accuracy of known attacks, robustness and resilience to novel attack patterns, and privacy preservation, using packet-level granular data. In the first stage, a global DNN model is collaboratively trained on distributed data, and the second stage adapts it to novel attack patterns. Our experiments on real-world intrusion datasets demonstrate the effectiveness of *FedNIDS* by achieving an average F1 score of 0.97 across distributed networks and quickly disseminating novel attack information within four rounds of communication.

CCS Concepts: • **Security and privacy** → **Network security**; **Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies** → **Machine learning algorithms**;

Additional Key Words and Phrases: Federated learning, network intrusion detection systems, distributed and private network data, packet-based NIDS, novel attack detection

ACM Reference format:
Quoc H. Nguyen, Soumyadeep Hore, Ankit Shah, Trung Le, and Nathaniel D. Bastian. 2025. FedNIDS: A Federated Learning Framework for Packet-Based Network Intrusion Detection System. *Digit. Threat. Res. Pract.* 6, 1, Article 4 (February 2025), 23 pages.
https://doi.org/10.1145/3696012

Quoc H. Nguyen and Soumyadeep Hore contributed equally to this research.
The views and conclusions expressed in this article are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.
This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0045, as well as the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Support Agreement No. USMA 21050.
Authors' Contact Information: Quoc H. Nguyen, University of South Florida, Tampa, Florida, USA; e-mail: nguyenq29@usf.edu; Soumyadeep Hore, University of South Florida, Tampa, Florida, USA; e-mail: soumyadeep@usf.edu; Ankit Shah (corresponding author), Indiana University, Bloomington, Indiana, USA; e-mail: ankit@iu.edu; Trung Le, University of South Florida, Tampa, Florida, USA; e-mail: tqle@usf.edu; Nathaniel D. Bastian, United States Military Academy, West Point, New York, USA; e-mail: nathaniel.bastian@westpoint.edu.

ACM 2576-5337/2025/2-ART4
https://doi.org/10.1145/3696012

1 Introduction

The proliferation of digital systems and the increased reliance on interconnected networks have elevated the significance of ensuring the security of computer systems and data. **Network intrusion detection systems (NIDS)** play a pivotal role in identifying potential threats and attacks within computer networks. These systems employ signature-based [16, 30] and anomaly-based [17, 51] techniques to detect attacks. With advancements in algorithmic knowledge and the availability of high-performance computing resources, **deep learning (DL)** methods have taken center stage in the development of state-of-the-art NIDS. DL-based NIDS are developed using both supervised [4, 8] and unsupervised [24, 52] training methods. With the assistance of labeled training data containing attack categories, the supervised learning paradigm has demonstrated superior performance in identifying previously known attacks when compared to the unsupervised learning approach. The type of data used for training DL-based NIDS can be broadly categorized into flow-based and packet-based data. While flow-based analysis provides valuable insights into network activity and has seen advancements through the incorporation of application metadata, our study focuses on leveraging the rich, granular information available in packet data. Packet-based analysis allows for more precise and timely detection of security incidents, complementing the high-level overview offered by flow data [11, 18, 20].

DL-based NIDS rely on the availability of large, consolidated datasets to learn the underlying patterns of benign and malicious network traffic. However, accumulating and consolidating data across different entities in distributed computer networks pose challenges in various situations: (i) Different branches within an organization, such as those in banks, hospitals, or the Department of Defense, often handle sensitive and confidential information that cannot be shared in a central location to train DL models. (ii) Many regions, like the European Union, and industries, such as healthcare under HIPAA regulations, enforce strict data privacy and security regulations that restrict the movement and sharing of data. (iii) Networks can exhibit significant variations in terms of their size and traffic patterns, and aggregating their data to train a centralized model may not necessarily result in superior detection accuracy across different network segments. (iv) Transmitting large volumes of data to a centralized server for DL model training may demand substantial bandwidth and introduce latency. Hence, this approach may not be well-suited for environments constrained by limited bandwidth and low-latency requirements, such as communication in military operations or real-time monitoring and transmission using **Internet of Things (IoT)** in an industrial setting. Another challenge in NIDS is the delayed identification of new attack patterns, such as zero-day and evasion attacks. This gap becomes particularly significant in a distributed setting where various entities cannot share novel attack data among themselves. This situation necessitates an adaptive mechanism for rapidly disseminating the knowledge from the affected participants. These challenges underline the need for alternative approaches in DL-based NIDS, which can address data privacy concerns, including regulatory constraints, and the variability in types of attacks experienced by the distributed entities while optimizing network intrusion detection.

The federated learning paradigm provides a compelling solution to the aforementioned challenges, empowering the collaborative training of models across multiple clients (distributed networks) without requiring centralization of data. This collaborative paradigm preserves data privacy and security and involves clients training their local models on their private data. Subsequently, the clients share the model updates in a secure and aggregated manner [13]. We introduce a novel framework called the **federated network intrusion detection system (FedNIDS)**, which leverages the synergy of federated learning and **deep neural networks (DNNs)** to enhance the accuracy and robustness of the NIDS against an evolving adversarial environment. *FedNIDS* aims to address the challenges in accurate detection of malicious network activities while upholding data privacy and security by training the DNNs on decentralized and privacy-sensitive packet-based data sources.

The primary contribution of this study is the development of the federated learning framework, *FedNIDS*, designed to operate in a distributed environment, utilizing granular packet-level data. The objective of this framework is to identify known attack patterns and adapt to detect emerging threats. It consists of two essential stages: federated DNN pre-training and federated novel attack fine-tuning. In the first stage, the framework trains a global DNN-based binary classifier by leveraging decentralized data from participating clients. The second stage involves fine-tuning the pre-trained global model from the first stage with novel attack samples, as observed by the client(s). Our study is the first to address the issue of **non-independently and identically distributed (non-IID)** data across different clients for packet-based NIDS. We employ publicly available benchmark network intrusion detection datasets for experimentation and evaluation. The results illustrate the robustness and resilience of our *FedNIDS* framework in a distributed environment when encountering various forms of attacks, including known and unknown threats, such as zero-day and adversarially perturbed evasion attacks. These findings collectively contribute to the advancement of NIDS technology, enhancing their ability to identify and mitigate a broad spectrum of network attacks, particularly in situations where network packet data among different clients is non-IID.

The structure of the remainder of the article is as follows: Section 2 reviews the existing literature on NIDS and federated learning. Section 3 outlines our methodological framework, *FedNIDS*. In Section 4, we delve into numerical experiments. Section 5 presents the results and offers an in-depth analysis. Finally, in Section 6, we draw conclusions from our study and suggest potential avenues for future research.

2 Related Work

We present the related literature review by dividing this section into the following subsections: (i) NIDS, (ii) federated learning, and (iii) federated learning in NIDS.

2.1 NIDS

NIDS play a critical role in safeguarding computer networks from unauthorized access and malicious activities. The entire NIDS literature can be broadly categorized into signature-based and anomaly-based NIDS. A signature-based NIDS relies on the matching of seen attack signatures. An anomaly-based NIDS tries to model the expected user behavior to detect any possible deviation from the normal user behavior. With advancements in computational technology and the availability of faster computational resources anomaly-based NIDS have gained more popularity recently. Anomaly-based NIDS rely on DL and statistical learning methods to identify malicious activities [34]. To enhance the effectiveness of NIDS, the integration of DNNs has emerged as a powerful approach. DNNs excel in capturing intricate relationships and complex patterns within network traffic data [22]. Their inherent ability to learn hierarchical features makes them well-suited for discerning subtle deviations in network behavior that may signify malicious activity [53]. Additionally, DNNs can dynamically adapt to evolving attack patterns by fine-tuning and retraining on new data, ensuring persistent accuracy in detecting emerging threats [60]. Their ability to model non-linear relationships proves especially advantageous for capturing intricate attack behaviors that could evade traditional methods [27]. By capitalizing on these strengths, DNNs offer a promising avenue to elevate the accuracy and robustness of anomaly detection within packet-based NIDS.

Depending on the training data required to train such anomaly-based NIDS, there exist two prominent approaches in NIDS design: flow-based and packet-based methods. We provide insights into these approaches, highlighting their strengths, weaknesses, and recent advancements. Flow-based NIDS operate by analyzing network traffic at the flow level, where a flow represents a sequence of packets between two network endpoints. These systems focus on extracting features from flow data to identify anomalies and attacks [40, 48, 50]. Flow-based techniques have seen advancements, such as the incorporation of application metadata, which enhances their utility in network security analysis. However, flow-based NIDS may face challenges in detecting certain types of attacks due to the aggregated nature of flow data [5]. Packet-based NIDS, on the other hand, inspect individual packets in the network traffic. They analyze packet contents, headers, and payloads to detect malicious patterns [11, 15]. Packet-based analysis allows for more precise and timely detection of security incidents, enabling in-depth packet inspection and effective content-based matching against known attacks [29, 38]. Furthermore, using raw packet features as training data alleviates the generalizability issue associated with flow-based NIDS relying on different Netflow tools [24]. Some of the advantages of using packet-based NIDS over flow-based are as follows: (1) Packet-based NIDS can perform in-depth packet inspection, allowing them to detect attacks with high precision [29]. (2) Flow-based NIDS provide a coarser view of network traffic, which makes it challenging to detect certain types of attack [5]. (3) Packet-based NIDS excel at content-based matching, making them very effective against known attacks [38]. (4) Flow-based NIDS rely on different Netflow tools (such as Wireshark and **Canadian Institute of Cybersecurity (CIC)** flowmeter) for compiling network packet information to obtain features for training, causing a loss of generalizability across different end users. Using raw packet features as training data alleviates the generalizability issue [24]. (5) Packet-based features enable real-time detection since flow-based features are obtained by analyzing the network communication after the flows are completed [18, 23].

2.2 Federated Learning

Federated learning has gained substantial attention as a transformative approach to decentralized **machine learning (ML)**. It enables model training across distributed data sources while preserving privacy and data locality. We provide an overview of key concepts, recent advancements, and notable research papers in the field of federated learning. McMahan et al. [36] define federated learning, in their seminal paper "Communication-Efficient Learning of Deep Networks from Decentralized Data," as an ML framework for training models across decentralized edge devices while keeping data local. This approach has evolved from the necessity to address privacy concerns and the increasing prevalence of edge and IoT devices. Federated learning typically consists of three main components: client devices (edge nodes), a central server, and federated learning algorithms. The client devices hold local data and perform local model updates, while the central server coordinates model aggregation and global updates [13]. The **federated averaging (FedAvg)** algorithm, introduced by McMahan et al. [36], is a fundamental algorithm that aggregates model updates from client devices to construct a global model. It involves iterative rounds of communication between clients and the central server. To ensure privacy, differential privacy mechanisms have been integrated into the federated learning algorithm. For example, techniques like DP-SGD [2] and secure aggregation [13] protect user data from exposure during model updates. Federated learning often assumes that client data distributions are **independent and identically distributed (IID)**. Research has focused on addressing the challenges posed by non-IID data, such as skewed distributions and non-uniform data access patterns among others [32]. Li et al. [33] modified the FedAvg algorithm to address non-IID data challenges by extending the standard federated optimization objective by adding a proximal/regularization term to the loss function. The algorithm is popularly known as FedProx [33], which encourages model updates from clients to be close to their previous models by effectively penalizing larger updates. The scope of federated learning's applications has transcended traditional boundaries, extending to domains like healthcare [59] and the IoT [57], accentuating its relevance for NIDS as well [31, 44].

2.3 Federated Learning in NIDS

Next, we discuss the use of federated learning in NIDS. In Table 1, we highlight some of the most recent and notable works employing federated learning in the NIDS domain. Agarwal et al. [3] provide an extensive and exhaustive review of the use of federated learning in NIDS. They also discussed the need for a federated learning paradigm, various types of intrusion detectors, and relevant ML approaches, along with potential issues. We observe that a major part of the literature focuses on building a federated learning framework for flow-based NIDS [6, 7, 9, 14, 25, 41, 49]. One of the major challenges in implementing federated learning for NIDS is data dimensionality as extracting flow-based features from different source datasets results in an inconsistent set of features across different clients. Packet-based implementation of the federated learning framework alleviates that problem since raw packet data features can be used directly for training, which remain consistent across the clients, irrespective of the source datasets. There exist very few works in the literature that employ a packet-based

Table 1. A Brief Literature Review of Federated Learning in NIDS

Paper Name	**Authors**	**Year**	**Algorithm**	**Global/ Local Model**	**Data Representation**	**Dataset(s)**
Fed-ANIDS: Federated Learning for Anomaly-Based NIDS [25]	Idrissi et al.	2023	FedProx	Autoencoders	Flow-based	USTC-TFC2016, CIC-IDS2017, CIC-IDS2018
Data-Efficient, Federated Learning for Raw Network Traffic [56]	Willeke et al.	2023	FedAvg	1dCNN, FcNN	Packet-based	UNSW-NB15, CIC-IDS2017
GowFed A Novel Federated Network Intrusion Detection System [9]	Belenguer et al.	2023	Fedavg with Gower distance	DNN	Flow-based	TON_IOT
Enhancing Privacy-Preserving Intrusion Detection through Federated Learning [6]	Alazab et al.	2023	FedAvg	DNN	Flow-based	NSL-KDD
Collaborative IDS for SDVN: A Fairness Federated Deep Learning Approach [14]	Cui et al.	2023	FedAvg with two-stage gradient optimization	1dCNN	Flow-based	KDD99, NSL-KDD
A Federated Learning-Based Approach for Improving Intrusion Detection in Industrial Internet of Things Networks [42]	Rashid et al.	2023	FedAvg	CNN, RNN	Flow-based	Edge-IIoTset
FLUIDS: Federated learning with Semi-Supervised Approach for IDS [7]	Aouedi et al.	2022	FedAvg	Autoencoders, DNN	Flow-based	UNSW-NB15
Towards Asynchronous Federated Learning-Based Threat Detection: A DC-Adam Approach [49]	Tian et al.	2021	DC-Adam	Denoising Autoencoders	Flow-based	CIC-IDS2017, IoT-23, MNIST
Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning [41]	Rahman et al.	2020	FedAvg	DNN	Flow-based	NSL-KDD
DIOT: A Federated Self-Learning Anomaly detection System for IoT [39]	Nguyen et al.	2019	FedAvg	GRU	Packet-based	Mirai

NIDS with the federated learning framework. Willeke et al. [56] use raw packet features extracted from the **packet capture (PCAP)** files, while Nguyen et al. [39] consider some extracted features like packet direction, port type, and **transmission control protocol (TCP)** flags, among others. Additionally, the latter also considers a sequence of packets between the security gateway and IoT devices for the detection task. Another important challenge in implementing federated learning in NIDS is the model divergence under non-IID data scenarios. The FedAvg algorithm, used in [56], performs well under IID assumptions. However, distributed clients in the NIDS setting experience non-IID data. The FedAvg algorithm is known to have underwhelming results in the presence of non-IID data across the different clients. The DC-Adam algorithm addresses this issue by using Taylor expansion to stabilize the loss function when minimizing it along with some warm-up procedures for parameter initialization [49]. Belenguer et al. in the GowFed paper address the model divergence issue by introducing a Gower dissimilarity [19] to limit the extent of the model updates. The FedProx algorithm [33] also performs well, in general, in the presence of non-IID data. Recently, some works have also explored poisoning attacks on federated learning-based NIDS [37, 58]. However, evaluating the resilience of federated learning-based NIDS classifier systems under novel attacks such as zero-day and evasion attacks remains unexplored.

To the best of our knowledge, there is a lack of existing literature on the development of a federated learning framework for NIDS using raw packet data in a non-IID data environment. There is also a notable absence of studies that have assessed and enhanced the robustness of federated learning-based NIDS in the face of emerging and unfamiliar attack scenarios. In this study, we aim to address these gaps by developing a federated learning framework for NIDS that is capable of recognizing known attacks and can be fine-tuned to quickly adapt to identify novel attacks across different clients.

3 Methodology

In this section, we present our methodology for the federated learning framework for packet-based NIDS (FedNIDS). We begin by discussing the preliminaries (Section 3.1), which include a mathematical overview of federated learning (Section 3.1.1), the non-IID data issue in federated learning (Section 3.1.2), and the problem statement (Section 3.1.3). We then introduce the proposed FedNIDS framework (Section 3.2), which consists of two main stages: federated DNN pre-training (Section 3.2.1) and federated novel attack fine-tuning (Section 3.2.2).

3.1 Preliminaries

3.1.1 Federated Learning: A Mathematical Overview. Federated learning in the client–server paradigm involves three main parties: (1) a multitude of silos (e.g., clients) denoted as $S = \{s_i\}_{i=1}^{N}$, where each s_i owns private datasets D_{s_i}; (2) a model owner, O, who initiates the process with a randomly initialized global model, represented by its parameters, w_G, and oversees its training; and (3) a secure aggregator Agg (e.g., a trusted server), positioned between O and S. Notably, O is oblivious to the specifics of D_{s_k} due to the implementation of a secure aggregation protocol by Agg [13]. Typically, **stochastic gradient descent (SGD)** is used by the silos to train their models (FedAvg [36]). Preceding the commencement of federated learning, w_G is initialized by O. During the tth aggregation round:

(1) Agg transmits w_G^t to a subset of silos $S_{sub} = \{s_i\}_{i=1}^{K}$, where $K \leq N$.
(2) Each silo s_i updates its local model from $w_{s_i}^{t-1}$ to $w_{s_i}^t = w_G^t$. Subsequently, s_i retrains $w_{s_i}^t$ through a predetermined number of local passes over its private dataset D_{s_i} and sends the retrained local model $w_{s_i}^t$ back to Agg.
(3) Agg aggregates the retrained local models to compute a new global model w_G^{t+1}.

The above process continues iteratively, until the global model w_G achieves the desired performance level or another predefined stopping criterion is met. The objective is to improve the global model's accuracy over successive rounds, while respecting privacy and security constraints.

3.1.2 Non-IID Data Issue in Federated Learning. A significant challenge within the realm of federated learning arises due to the presence of non-IID data across different silos [26, 32]. Non-IID data refer to data that do not share the same statistical characteristics among various silos. This non-IID nature of data can significantly impact the accuracy of the federated learning algorithm. The underlying reason lies in the distinctiveness between the distribution of each local dataset within different silos and the global distribution. Consequently, the objectives of the local models within each silo can be misaligned with the overarching goal of achieving global optimality. This misalignment leads to a *drift* in the local updates [28, 54, 55].

To simplify, during the local training phase within each silo, the models strive to improve their respective local objectives. However, these local objectives may lead the models to local optima that differ considerably from the desired global optima. Moreover, when substantial local updates occur, such as an extensive number of local training epochs [33], the average model derived from combining these local models can also deviate significantly from the global optima. As a result, the final global model's accuracy is notably compromised, particularly when compared to scenarios where data is IID. This challenge is visually illustrated in Figure 1, which vividly captures the hurdle faced by FedAvg when dealing with non-IID data scenarios involving silos. To elaborate further, consider the scenario where the global optimum w_G^* is close to the local optima w_1^* and w_2^* under the IID setting. This results in the averaged model w^{t+1} being close to the global optimum. However, in non-IID settings, when w_G^* is distant from w_1^*, the averaged model w^{t+1} can deviate significantly from the global

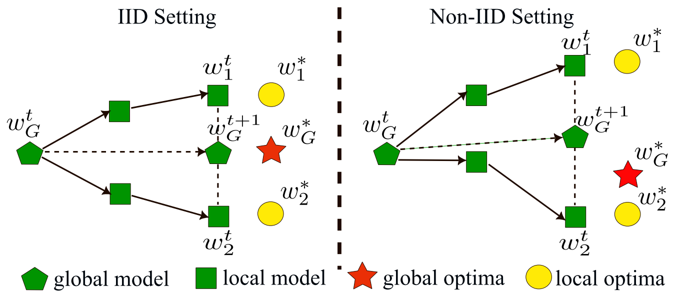


Fig. 1. Example of a drift under the non-IID setting.

optimum. Effectively tackling this challenge requires designing robust algorithms tailored to handle non-IID data present across different silos.

3.1.3 Problem Statement. Consider N clients or silos denoted as $S = \{s_i\}_{i=1}^{N}$, each possessing a local dataset D_{s_i} containing network traffic samples. The objective is to develop a global classification model that can accurately identify malicious samples within the network traffic data from these distributed datasets, while upholding data privacy and security. While flow-based analysis has its merits in providing a high-level overview of network activity, our study focuses on packet-based analysis to address the challenges of non-IID data in a federated learning setting for NIDS. We pose the problem as follows: Given the distributed datasets D_{s_i} develop a federated learning-based DNN model that optimizes a global objective function while respecting the privacy constraints of individual clients. Mathematically, we aim to minimize the global objective function $\mathcal{L}(w)$, which is defined as

$$\arg\min_{w} \mathcal{L}(w) = \sum_{i=1}^{N} \frac{|D_{s_i}|}{|D|} \mathcal{L}_i(w).$$

In the above equations:

— $|D_{s_i}|$ represents the size (number of samples) of the local dataset for a particular silo s_i.
— $|D|$ represents the total size (number of samples) of the entire dataset, which comprises the combined data from all silos.
— $\mathcal{L}_i(w)$ is the local objective function for silo s_i, measuring the goodness of fit or the loss of the global model w on the data specific to silo s_i. It is defined as $\mathcal{L}_i(w) = \mathbb{E}_{x;y\sim D_{s_i}}[\ell_i(w;x;y)]$.

The optimization problem, $\arg\min_w \mathcal{L}(w)$, seeks to find the global model parameters w that minimize the objective function $\mathcal{L}(w)$ ensuring effective performance across all data sources while respecting privacy constraints in federated learning. This is achieved by aggregating local losses $\mathcal{L}_i(w)$ from each source, weighted by their data contributions, to formulate $\mathcal{L}(w)$. Minimizing $\mathcal{L}(w)$ aims to create a model that balances performance across all sources, promoting effectiveness on the combined data from multiple parties, the core objective in distributed data scenarios. Our research aims to develop a federated learning-based DNN model that can effectively detect attacks across different silos and efficiently adapt to novel attack patterns when new data becomes available.

3.2 Federated Learning Framework for Packet-Based Network Intrusion Detection System (*FedNIDS*)

To attain the aforementioned research objective, we propose a generalized federated learning framework, named *FedNIDS*, to enhance both the robustness and the performance of federated models when learning from decentralized data with statistical heterogeneity in NIDS. Our framework comprises two stages: a federated DNN pre-training stage and a federated novel attack fine-tuning stage, as shown in Figure 2 and Algorithm 1. During the federated DNN pre-training stage, the model exploits knowledge from decentralized data by pre-training with

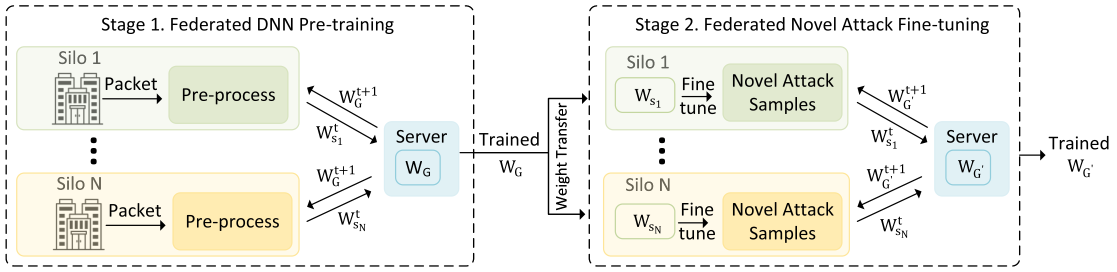


Fig. 2. Overview of the federated learning framework. In the pre-training stage (left), the packet data are processed to learn representations in each silo. The pre-training process consists of three steps and ends when it reaches the maximum communication rounds T. At round t, (1) Each silo $S_i (i \in \{1, ..., N\})$ trains its local DNN with local data; (2) Silo s_i uploads the weights $w_{s_i}^t$ to the central server; (3) The server produces a global DNN model with weights $w_{s_i}^{t+1}$ via model weights averaging and broadcasts the global model back to each local silo. In the fine-tuning stage (right), the final pre-trained global W_G from the first stage is used to initialize each local DNN. The W_G is then fine-tuned with new attack samples. End-to-end federated fine-tuning is performed on the novel attack samples in each local silo to get the desired model $W_{G'}$.

pre-processing procedure in a distributed setting. In the federated novel attack fine-tuning stage, the knowledge is transferred from the previous stage to the target task by fine-tuning the federated models with novel attack samples.

Algorithm 1 presents the pseudocode for *FedNIDS*. The algorithm consists of two main stages: federated DNN pre-training and federated novel attack fine-tuning. In the federated DNN pre-training stage (lines 2–9), the algorithm iteratively trains the global model by aggregating the local model updates from the participating silos. The local training is performed on each silo's data (lines 12–18), where the local models are updated using their respective datasets. The model updates from each silo are then aggregated to update the global model (line 8). In the federated novel attack fine-tuning stage (lines 19–23), the algorithm adapts the pre-trained global model to detect novel attack patterns. When a silo encounters novel attack samples, the fine-tuning process is triggered. The local models are updated using these novel attack samples (lines 20–22), allowing the global model to learn and adapt to new attack patterns. In the following Sections 3.2.1 and 3.2.2, we provide detailed descriptions of federated DNNs pre-training and federated novel attack fine-tuning stages, respectively.

3.2.1 *Federated DNNs Pre-Training.*

The first stage of our proposed *FedNIDS* framework focuses on enhancing the global model's initialization through federated DNN pre-training. The goal is to develop robust initialization of local DNNs at each silo that can effectively capture relevant features from their respective datasets. This stage reduces the need for extensive communication rounds between the clients and the server, thereby enhancing the efficiency of the federated learning process.

The pre-training process occurs in a distributed setting and involves the following steps, repeated for a maximum of T communication rounds:

(1) Pre-Processing: Numerical features are extracted from the raw PCAP data files. The pre-processing step is kept simple and generalizable across different organizations. As discussed earlier, flow-based features raise the issue of inconsistent feature sets across organizations and hinder generalizability. Hence, we use the raw PCAP files to extract numerical features. The dataset creation tool is adopted from [23]. A comprehensive guide to extracting and labeling the PCAP files are as follows:
   —Use Scapy [12] and dpkt [47] to parse different header and segment information from the raw PCAP file.
   —Use the 6-tuple information containing source IP address, destination IP address, source port, destination port, protocol, and epoch time to differentiate between benign and malicious packets.

Algorithm 1: FedNIDS

```
Input: Number of communication rounds $T$, number of local epochs $E$, number of silos $N$, local data set $D_{s_i}$, learning rate $\eta$
Output: Global models $w_G, w_{G'}$
Server executes:
1:  Initialize $w_G^0$;
2:  for each round $t = 0, 1, ..., T-1$ do
3:      Sample a set of clients $S^t$;
4:      $n \leftarrow \Sigma_{i \in S^t} |D_{s_i}|$;
5:      for each silo $s_i \in S^t$ in parallel do
6:          Send the global model $w_G^t$ to silo $s_i$;
7:          $\Delta w_{s_i}^t \leftarrow$ LocalTraining($s_i, w_G^t$);
    ▷ Stage 1: Federated DNN Pre-training
8:      $w_G^{t+1} \leftarrow \Sigma_{i=1}^{N} \frac{|D_{s_i}|}{n} \Delta w_{s_i}^t$;
    ▷ Stage 2: Federated Novel Attack Fine-tuning
9:      $w_{G'}^{t+1} \leftarrow \Sigma_{i=1}^{N} \frac{|D_{s_i}|}{n} \Delta w_{s_i}^t$;
Client executes:
10: $\mathcal{L}(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y) + \frac{\mu}{2} ||w - w_t||^2$;
11: LocalTraining($s_i, w_G^t$);
    ▷ Stage 1: Federated DNN Pre-training
12: Sample batches $\mathbf{b} = \{x, y\}$ from local data $D_{s_i}$;
13: $\mathbf{b}' \leftarrow$ Pre-process($\mathbf{b}$);
14: $w_{s_i}^t \leftarrow w_G^t$;
15: for each local epoch $k = 1, ..., E$ do
16:     for each batch $\mathbf{b}'$ do
17:         $w_{s_i}^t \leftarrow w_{s_i}^t - \eta \nabla \ell(w_{s_i}^t; \mathbf{b}')$;
18: $\Delta w_{s_i}^t \leftarrow w_G^t - w_{s_i}^t$;
    ▷ Stage 2: Federated Novel Attack Fine-tuning
19: for each local epoch $k = 1, ..., E$ do
20:     for each batch $\mathbf{b}'$ do                ▷ Updated batch $\mathbf{b}'$ with novel attack samples
21:         $w_{s_i}^t \leftarrow w_{s_i}^t - \eta \nabla \ell(w_{s_i}^t; \mathbf{b}')$;
22: $\Delta w_{s_i}^t \leftarrow w_{G'}^t - w_{s_i}^t$;
23: return $\Delta w_{s_i}^t$ to the Server;
```

— Extract only forward packets resulting in a dataset of unidirectional communication, i.e., only extract packets that are being sent from identified source IP addresses excluding any response from the server.
— To alleviate inherent bias in training ML models, we remove some header information (ethernet header, source and destination IP address, source and destination port information).
— Set a maximum feature length for all extracted packets. However, the number of bytes in a packet may vary greatly. To counter this zero-padding is used that helps in maintaining a standard structure of the data.

- Convert raw packet information from hexadecimal to decimal. Each byte value contains a combination of two hexadecimal numbers, ranging from 00 to FF. The hexadecimal numbers are converted to decimal numbers between 0 and 255 and subsequently normalized for efficient machine computation.

(2) Each silo s_i (where $i \in \{1, \ldots, N\}$) trains its local DNN-based classifier with its own pre-processed data for E local epochs.
(3) Silo s_i uploads the weights $w^t_{s_i}$ of its DNN to the central server.
(4) The central server aggregates the uploaded weights using a model weights averaging technique and produces an updated global DNN model with weights w^{t+1}.
(5) The server broadcasts the updated global model to each local silo s_i.

In addition to the pre-training stage, our *FedNIDS* framework incorporates the FedProx technique [33] to enhance the local objective function based on the FedAvg algorithm. This technique introduces an $L2$ regularization term in the local objective function to limit the distance between the local model and the global model. This regularization term effectively controls the size of local updates, preventing the local models from diverging too far from the global optima. The effectiveness of the regularization term in handling non-IID data has been demonstrated in various studies [28, 33]. These studies have shown that FedProx achieves better performance and faster convergence compared to FedAvg in the presence of non-IID data. A hyper-parameter μ is introduced to adjust the weight of the $L2$ regularization term. While the modification to FedAvg is lightweight and easy to implement, it introduces some additional computation overhead. However, it does not introduce extra communication overhead. This approach requires fine-tuning of μ to achieve superior accuracy. If μ is too small, the regularization term might have minimal impact. Conversely, if μ is too large, the local updates become very small, potentially leading to slower convergence speed.

This stage concludes when the maximum communication rounds T are reached. The global model w_{T+1} obtained at the end of the pre-training stage serves as a robust initialization for the subsequent novel attack fine-tuning stage. The combination of pre-processing and federated DNN pre-training enhances the quality of the learned features and the global model's ability to capture important characteristics of network traffic data, improving the overall effectiveness of the *FedNIDS* framework.

3.2.2 Federated Novel Attack Fine-Tuning.

In the second stage of our FedNIDS framework, we perform federated novel attack fine-tuning by utilizing the knowledge gained during the pre-training stage to the target task of network intrusion detection. It is important to note that the fine-tuning step differs from periodic retraining of the global model. Periodic retraining involves regularly updating the model with new data from the participating silos to maintain its performance and relevance over time. This process typically includes both benign and malicious traffic patterns and is performed at regular intervals. In contrast, the fine-tuning step specifically focuses on adapting the model to novel attack patterns that were not present in the initial training data. It allows for a rapid response to emerging threats by quickly updating the model with novel attack samples. The fine-tuning step can be triggered whenever novel attack patterns are detected, while periodic retraining can be performed at regular intervals to incorporate a broader set of newly observed data. The fine-tuning step provides an additional layer of adaptability to address novel attack patterns in a timely manner.

The fine-tuning process occurs as follows:

(1) The trained global model w_{T+1} from the pre-training stage is used to initialize each local DNN at the respective silos.
(2) Each silo s_i continues to contribute to the learning of the global model with their own local datasets D_{s_i}, which include newly experienced novel attack samples.
(3) The fine-tuning involves updating the DNN's parameters, w, using an optimization algorithm such as SGD to minimize the loss on the novel attack samples.

This federated novel attack fine-tuning stage effectively adapts the trained global model to the specific characteristics of each silo's dataset, enabling the model to detect new attack patterns that may arise in the future. The combination of the two stages within the *FedNIDS* framework aims to enhance the model's accuracy, robustness, and generalization capabilities for network intrusion detection.

4 Experiments

In this section, we outline the numerical experiments performed to evaluate our *FedNIDS* methodology. We first discuss the experimental data followed by the description related to the non-IID data silo creation. Subsequently, we discuss the hyper-parameter configurations employed for *FedNIDS*. We evaluate the performance of our approach by comparing it against two other federated learning approaches from the literature: (1) FedAvg [56], which was recently utilized for NIDS and (2) FedAdam [43], an extension of the Adam optimizer to the federated setting, which uses moment estimates to adapt the learning rates for each model parameter. The experimentation was carried out employing a computing system featuring the NVIDIA GeForce RTX 3070 graphics processing unit with 12 GB of video memory and a substantial 128 GB of DDR4 RAM operating at a speed of 3,200 MHz. We built a Python-based simulator to emulate the federated learning process, in which the total available computational resources were equally divided amongst the silos.[1]

4.1 Data Description

A federated learning framework in the context of NIDS offers significant advantages in preserving both data privacy and security, all the while maintaining data ownership and control. Nevertheless, obtaining real-world organizational traffic data for research is a daunting task for the same reasons. Hence, researchers turn to publicly available benchmark datasets as a practical substitute. In this study, we use raw PCAP files from two publicly available intrusion detection datasets, namely, CIC-IDS2017 [45] and CIC-IDS2018 [46] to extract benign and attack communications. We select CICIDS datasets over some popular yet older datasets like NSL-KDD and KDD-CUP due to the following reasons: (i) they provide a more pragmatic representation of modern network traffic [21] and (ii) they provide easy accessibility to the raw PCAP files, thereby reducing the dependency on flow-level data. These datasets are accumulations of network activities recorded over several days. The details can be found on the CIC Web site [1], and the threat model for the conducted attacks is outlined in [46]. A brief description of the PCAP files, in terms of their size and contents, is provided in Table 2.

We follow the process in [23] to obtain the packet information from the PCAP files. We analyze both the header and payload sections of the packet. The header section includes information such as **time to live (TTL)**, window size, fragmentation, total length, acknowledgment number, and flags, among others. The payload section contains the actual data being communicated. These features can provide insights into various attacks. For example, unusually large packet sizes may indicate attempts to exploit vulnerabilities by overflowing buffers or injecting malicious payloads. Additionally, anomalies in packet headers, such as forged or spoofed TTL values, can indicate attempts to evade detection.

We exclude Botnet attack packets from the CIC-IDS2017 dataset during the initial phase of the experimentation so that we can introduce Botnet attack as a zero-day attack to evaluate the resilience of our framework. In our experimentation, we focus on analyzing unidirectional forward packets, which are packets sent from source IP addresses, excluding any responses from the server. This approach allows us to concentrate on detecting attacks originating from specific sources, enabling effective attribution and facilitating targeted security measures. By considering only the forward packets, as in [23], simplify the data pre-processing and feature extraction process, focusing on the relevant information carried in the packets sent by potential attackers.

In a TCP packet, the maximum byte count is 1,554, distributed as follows: 1,460 bytes for the application layer, 60 bytes for the transport layer (20 for the header and a maximum of 40 for options), 20 bytes for the internet

[1]We have made the source code accessible at the following URL: https://github.com/quocnh/Fed_NIDS

Table 2. Description of CIC-IDS2017 and CIC-IDS2018 PCAP Files

CIC-IDS2017			**CIC-IDS2018**		
Day/Date	**File Size**	**Activity**	**Day/Date**	**File Size**	**Activity**
Monday/ 3 July 2017	10 GB	Benign	Wednesday/ 14 February 2018	147 GB	SSH/FTP Patator and Benign
Tuesday/ 4 July 2017	10 GB	Brute Force, and Benign	Thursday/ 15 February 2018	57.8 GB	DoS Goldeneye, DoS Slowloris, and Benign
Wednesday/ 5 July 2017	12 GB	DoS, and Benign	Friday/ 16 February 2018	460 GB	DoS Slowhttptest, DoS Hulk, and Benign
Thursday/ 6 July 2017	7.7 GB	Web Attack, Infiltration, and Benign	Wednesday/ 21 February 2018	97.5 GB	DDoS and Benign
Friday/ 4 July 2017	8.2 GB	Botnet, Port Scan, DDoS, and Benign	Thursday/ 22 February 2018	110 GB	Web Attack and Benign
-	-	-	Friday/ 23 February 2018	65.9 GB	Web Attack and Benign
-	-	-	Wednesday/ 28 February 2018	73.1 GB	Infiltration and Benign

layer, and 14 bytes for the Ethernet layer. To prevent potential bias in the learning process of DL models, certain information is excluded, such as MAC addresses, as well as source and destination IP addresses. Consequently, we are left with 1, 525 features, all of which are kept in our dataset to ensure segment information is not excluded. The hex payload values in these features are converted to decimal values between 0 and 255, and later normalized to values between 0 and 1.

To evaluate the robustness of the *FedNIDS* framework, we also subject it to adversarial examples generated by modifying different attack packets encountered in the CICIDS datasets. We created two different toolchains to generate these evasion attacks with both constrained and unconstrained perturbations. Toolchain 1 was developed using a DRL approach from literature [23], which makes subtle perturbations such that the functionality and maliciousness of these packets remain intact (constrained). Table 3 shows the features that we considered for perturbations using Toolchain 1 and their resulting impact on other features. Toolchain 2 was developed using a heuristic approach, in which we made perturbations by making a random selection of features (unconstrained). We then methodically increased the feature values by a constant value as in [18]. We considered a white-box setting for the generation of the adversarial samples to test the worst-case scenario for the NIDS. In this setup, we considered a DNN model with the same architecture as used by the global model in the federated learning framework. The samples that evaded the DNN model were collected and stored for each of these toolchains.

4.2 Experimental Setup

Next, we discuss the segregation of the data into four different silos to evaluate the effectiveness of the *FedNIDS* framework. In the FL-NIDS literature, most of the research studies focus on creating a global model with the IID assumption. In a realistic scenario, where we want to create a global NIDS with data coming from different organizations, often it is very difficult to adhere to the IID assumption. To counter this, we partition the

Table 3. Description of Perturbations Used to Generate Adversarial Examples

Perturbation	Description	Impact
Modifying fragmentation	Fragmentation bytes can be modified from do not fragment to do fragment/more fragment or vice versa.	Directly impacts fragmentation bytes (byte numbers 7, 8 of the IP header) indirectly impacts TTL value (byte number 9 of the IP header) and IP checksum (byte numbers 11, 12 of the IP header).
Modifying TTL	Increasing/decreasing the TTL value by a small integer between 1 and 255.	Directly impacts the TTL byte (byte number 9 of the IP header) and indirectly impacts IP checksum (byte numbers 11, 12 of the IP header).
Modifying window size	Increasing/decreasing the window size value by a small integer between 1 and 65335.	Directly impacts window size bytes (byte number 15, 16 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header).
Adding/modifying maximum segment size	Only limited to SYN and SYN-ACK packets. The MSS value is limited between 0 and 65335.	Directly impacts TCP options (bytes after 20 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header).
Adding/modifying window scale value	Only limited to SYN and SYN-ACK packets. The MSS value is limited between 0 and 14.	Directly impacts TCP options (bytes after 20 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header).
Adding segment information	Add dead bytes/most commonly occurring segment information.	Directly impacts the TCP segment, and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header).

extracted data based on the shards partition used in [36], a popular method for segregating data into non-IID partitions/shards. There exist multiple strategies for shard partitioning, namely, range-based, hash-based, key-based, and round-robin. In this research, we employ hash-based shard partitioning, which distributes data across shards based on a hash function applied to a specific attribute. The selected attribute is the label of the packet data. We created silos based on the selected partitioning strategy. Table 4 illustrates how the various malicious and benign samples from the two datasets are segregated into different silos. To maintain data integrity and ensure the reliability of our model's performance, we partition the data within each silo into separate training, validation, and testing sets. The training set is utilized to update the model parameters during the federated learning process, while the validation set is used for hyper-parameter tuning and model selection. The testing set is kept entirely separate and is exclusively used for the final evaluation of the trained model. This stringent data segregation protocol ensures that there is no overlap between the data subsets, preventing any information leakage that could lead to overly optimistic performance estimates.

The silos are carefully constructed to preserve the non-IID nature of the data across them and to demonstrate the attack detection efficacy of our framework without explicit data sharing. It is also important to note that some silos experience attacks that no other silos encounter in this setup. As discussed in the previous sections, having non-IID data across different silos poses a major challenge to the traditional federated learning frameworks, which our methodology aims to overcome. To exhibit the non-IID nature of the partitioned silos, we perform pairwise hypothesis testing. Particularly, to assess whether two samples have the same underlying distribution,

Table 4. Data Description for Different Silos

Silo	**Packet Type**	**Source Data set**	**Number of Forward Packets**		
			Train	**Validation**	**Test**
Silo 1	Port Scan, DoS, DDoS, and Benign	CIC-IDS2017	133,906	15,000	33,476
Silo 2	Brute Force, Infiltration, Web Attack, and Benign	CIC-IDS2017	110,585	15,000	30,146
Silo 3	DoS, Brute Force, and Benign	CIC-IDS2018	102,947	15,000	25,736
Silo 4	DDoS, Infiltration, Web Attack, and Benign	CIC-IDS2018	102,324	15,000	25,581

Table 5. Percentage of Non-IID Samples between Different Silos

Silo	**Silo 1**		**Silo 2**		**Silo 3**		**Silo 4**	
Statistical Tests	**Rank Sum Test**	**K-S Test**	**Rank Sum Test**	**K-S Test**	**Rank Sum Test**	**K-S Test**	**Rank Sum Test**	**K-S Test**
Silo 1	-	-	65.02	90.80	90.52	91.54	93.62	94.45
Silo 2	65.02	90.80	-	-	90.22	98.57	93.20	97.84
Silo 3	90.52	91.54	90.22	98.57	-	-	87.63	88.14
Silo 4	93.62	94.45	93.20	97.84	87.63	88.14	-	-

we perform the **Kolmogorov-Smirnov (K-S)** test [10]. We also perform the Mann-Whitney rank sum test [35] to compare two independent samples to determine if they have different distributions, or if one tends to have larger values than the other. Table 5 exhibits the pairwise K-S test and rank sum test results among different silos of data. We perform the tests on a small subset of 500 random samples drawn from each silo of packet data that has some payload information. Each sample from each silo is compared against all other samples from all other silos. The numbers reported in Table 5 are the percentage of times the K-S test/the rank sum test result indicated toward two samples being different. The high percentages reported in Table 5 indicate a significant difference between the data distributions of the silos. The percentage values represent the proportion of pairwise comparisons where the statistical tests (K-S test and rank sum test) rejected the null hypothesis of identical distributions. Higher percentage values suggest a greater degree of non-IID nature between the silos. In our experiments, we were able to reject the null hypothesis of the rank sum test on an average 86.70% of the total conducted experiments and the K-S test null hypothesis got rejected 93.55% of the total conducted experiments. From Tables 4 and 5, we can determine that the data partitions we crafted are of non-IID nature.

To demonstrate the effectiveness of the federated learning framework, we consider a setup with four clients, denoted as $N = 4$ silos, actively participating in training the global model during each communication round. Creating the training dataset for each silo poses a significant challenge due to the inherent imbalance across various attack and benign packet types. To address this, we employed a strategy wherein, for each silo, a subset of attack packets was randomly selected from the available attack types. Concurrently, we downsized the pool of benign packet samples to equal the total number of attack samples. This procedure was replicated across all silos in the experiment. For example, in Silo 1, the abundance of DDoS packets far outweighs that of other attack types, while DoS attack packets are relatively scarce. To rectify this, we randomly selected a subset of DDoS

Table 6. Hyper-Parameter Values for the DNN Model

Hyper-Parameter	**Grid Search**	**Optimal Value**
Regularization (μ)	0.001, 0.005, 0.01, 0.05, 0.1	0.005
Dropout rate	0.1, 0.2, 0.3, 0.4, 0.5	0.3
Architecture	[512, 256, 128, 64], [256, 128, 64], [128, 64]	[512, 256, 128, 64]
Activation function	ReLU, Tanh, Sigmoid	ReLU
Learning rate (η)	0.1, 0.01, 0.001, 0.0001	0.01
Batch size (b)	32, 64, 128, 256	32
Epochs (E)	5, 10, 15, 20	5

and Port Scan packets, ensuring they match the number of DoS packets. Subsequently, we balanced the benign packets by randomly downsampling them to align with the total number of attack packets.

The choice of hyper-parameters plays a crucial role in achieving optimal performance. The architecture and hyper-parameter values were selected based on literature studies [9, 41], experimental results, and computational constraints. It is important to note that the hardware and GPU cards can significantly influence the search for optimal hyper-parameters. With additional computational resources, a more extensive grid search, incorporating larger grid sizes and a greater number of hyper-parameters, could be conducted. The tradeoff lies in the compute time required for such a comprehensive search. In our experiments using the system described earlier in this section, we employed grid search cross-validation to identify the best hyper-parameters. The maximum grid size across all tuned hyper-parameters was five.

Techniques such as L2 regularization, dropout, early stopping, and architecture search were employed to avoid overfitting. Table 6 displays the grid of hyper-parameter values searched, along with the optimal values selected for the DNN model. The following parameter values were chosen based on extensive experimentation: number of rounds ($T = 5$) governs the frequency of communication rounds between clients and the central server, allowing for iterative updates; number of local epochs ($E = 5$) dictates the number of times each client refines its local model on its respective dataset, influencing the granularity of updates; a batch size of 32 samples ($b = 32$) determines the number of data samples processed together in each iteration, balancing parallelization and memory constraints; learning rate ($\eta = 0.01$) regulates the step size of parameter updates, influencing the convergence speed; and the regularization parameter ($\mu = 0.005$) helps mitigate overfitting by adding a penalty term based on the model weights. These values were meticulously chosen to strike a balance between convergence speed, model generalization, and available computational resources. In addition to hyper-parameters, we tailored the architecture of our DNN to best suit our objectives. The *input_size* was set to $1,525$, reflecting the number of features in the dataset and defining the input data's dimensionality. The network architecture encompassed four hidden layers with sizes of 512, 256, 128, and 64 units respectively, enabling the model to progressively learn abstract features from the input data. The activation function was **rectified linear unit (ReLU)**. The *num_classes* was set to 2, signifying the model's task of predicting between two output classes: benign or malicious network traffic. Consistency in architecture and hyper-parameter values was maintained across all algorithms considered for comparison.

5 Results

In this section, we present the experimental results and conduct an analysis of them. First, we delve into the performance of the global *FedNIDS* model against the various types of attacks observed at different silos. Next, we assess the framework's adaptability to novel attacks, encompassing previously unseen attack types and adversarially perturbed evasion attacks.

Table 7. Performance Comparison of FedNIDS with Other Approaches

Silo	Class	FedAvg			FedAdam			Centralized			FedNIDS		
		Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
Silo 1	Attack	1.00	0.97	0.98	0.99	0.99	0.99	0.99	0.98	0.99	**1.00**	**0.99**	**0.99**
	Benign	0.95	1.00	0.98	1.00	1.00	1.00	0.97	0.99	0.98	1.00	1.00	1.00
Silo 2	Attack	1.00	0.96	0.98	0.99	0.99	0.99	0.99	1.00	0.99	**1.00**	**0.99**	**0.99**
	Benign	0.95	1.00	0.97	1.00	1.00	1.00	0.99	1.00	0.99	0.96	0.93	0.95
Silo 3	Attack	1.00	0.68	0.80	0.99	0.65	0.78	0.99	0.90	0.95	**1.00**	**0.95**	**0.96**
	Benign	0.75	1.00	0.86	0.74	1.00	0.85	0.91	0.99	0.95	0.97	1.00	0.96
Silo 4	Attack	1.00	0.67	0.80	1.00	0.64	0.78	0.99	0.90	0.94	**1.00**	**0.96**	**0.94**
	Benign	0.75	1.00	0.86	0.73	1.00	0.84	0.90	0.99	0.94	0.98	0.97	0.95

The bold values represent top scores.

5.1 Performance of the Global FedNIDS Model

We evaluate the performance of the trained global *FedNIDS* model at each silo and compare it to the FedAvg algorithm [56], the FedAdam algorithm [43], and a centralized approach, where the same DNN model is trained using the combined data from all four silos. In the centralized approach, we train the model by consolidating the training sets from each of the silos and evaluating it on the same testing data samples as the other approaches. Table 7 presents precision (prec), recall (rec), and F1 score metrics for both attack and benign classes. We report the performance of each approach in accurately classifying attack and benign samples for each silo. Notably, the global *FedNIDS* model exhibits the strongest performance across all the silos and among all the methods, achieving the highest precision and recall values for attack samples.

For Silos 1 and 2, *FedNIDS* demonstrates remarkable precision, recall, and F1 score values for both attack and benign classes. Specifically, for the attack class, the model achieves precision, recall, and F1 score values greater than or equal to 0.99, indicating its ability to accurately identify attack samples with minimal false positives or false negatives. Similarly, for the benign class, the model exhibits high precision and recall values, resulting in F1 scores of 1.00 and 0.95, respectively. This showcases the model's ability to maintain accurate classification of benign samples, while effectively detecting attacks.

In Silos 3 and 4, where attack patterns are more complex and diverse, the global *FedNIDS* model maintains strong performance. For the attack class in Silos 3 and 4, the model achieves precision values of 1.00 and recall values greater than or equal to 0.95, with F1 scores at or above 0.95. This reflects the model's ability to effectively classify various attack samples. Furthermore, in Silo 3, the model demonstrates high precision (0.97) and recall (1.00) for the benign class, resulting in an F1 score of 0.96. In Silo 4, the model achieves a precision of 0.98 and a recall of 0.97 for the benign class, yielding an F1 score of 0.95. These outcomes underline the robustness and adaptability of the global *FedNIDS* model to different types of samples in the silos' datasets.

An intriguing observation is that *FedNIDS* attains a similar level of performance as the centralized approach and outperforms other federated learning algorithms, such as FedAvg and FedAdam. The results in Table 7 show that *FedNIDS* surpasses FedAvg's performance, particularly in Silos 3 and 4, where the attack patterns are more diverse. The superior performance of *FedNIDS* compared to FedAvg can be attributed to its ability to handle non-IID data distributions more effectively. *FedNIDS* adapts to the unique characteristics of each silo's network environment and captures the intricate patterns in the network traffic data. On the other hand, FedAdam demonstrates comparable performance to *FedNIDS* in Silos 1 and 2 but falls short in Silos 3 and 4. The adaptive moment estimation used in FedAdam helps in accelerating the convergence and improving the model's ability

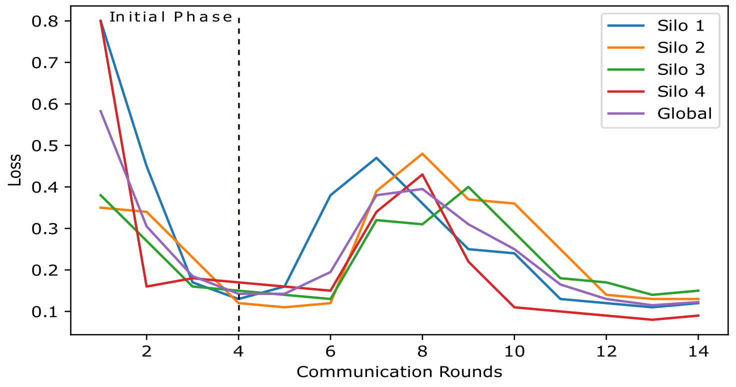


Fig. 3. FedNIDS adaptation to emerging threats.

to handle non-IID data to some extent. However, the results suggest that FedAdam may not be as effective as *FedNIDS* in capturing the complex attack patterns in Silos 3 and 4, where the attack samples are more diverse.

The superior performance of *FedNIDS* compared to FedAvg and FedAdam highlights the potential of our proposed framework in achieving accurate and robust detection outcomes in a federated learning setting. This comparative analysis underscores the importance of selecting appropriate federated learning algorithms and techniques tailored to the specific requirements of the NIDS domain. The ability of *FedNIDS* to maintain high performance across all silos, surpassing the centralized approach and other federated learning algorithms, demonstrates its potential for real-world deployment in decentralized network environments. By leveraging the strengths of federated learning while addressing the limitations of existing algorithms, *FedNIDS* offers a promising solution for accurate and responsive network intrusion detection in a privacy-preserving manner. Figure 3 further illustrates the convergence of the loss values at all the silos by the end of the fourth round, indicating the effectiveness of *FedNIDS* in learning from distributed data and adapting to the unique threat landscapes of each silo.

5.2 Adaptation to New Threats

Network security is a dynamic field, demanding adaptive and robust solutions to counter emerging threats. In this experiment, we evaluate the capability of *FedNIDS* to adapt to new, previously unseen attacks. To perform this evaluation, we introduce held-out botnet attack samples, which had not been encountered by any of the silos previously. Silo 1 is exposed to these novel attack samples. It is important to note that, at the time of this experiment, all silos were employing the trained global model, whose performance had been detailed in the previous experiment after four rounds of communication (refer to Figure 3).

This experiment aims to demonstrate the efficacy of the second stage of the *FedNIDS* framework. Figure 3 (after round four) illustrates the subsequent rounds of updates between the clients and the server (as outlined in Algorithm 1), starting from the moment Silo 1 encountered the botnet attack samples. Initially, it is evident that the trained global model struggles with detecting the botnet attack pattern from Silo 1. This challenge is reflected in the higher loss values, indicating suboptimal performance during the early rounds. Nevertheless, as the training progresses with additional rounds, the global model gradually learns to recognize the botnet attack pattern through the weight updates originating from the Silo 1 model.

After approximately four additional rounds, the model exhibits a significant reduction in loss across all silos, signifying a successful adaptation to the detection of botnet attack samples. The global model curve, which represents the average performance of all silos, illustrates the collective learning process. We conducted multiple trials involving various previously unseen attacks from the CICIDS datasets and consistently observed that

FedNIDS could adapt to new threats within approximately four rounds, on average. This experiment effectively demonstrates that the *FedNIDS* framework can adeptly respond to new threats, providing an efficient solution for sharing knowledge (rather than raw data) about a new attack experienced at one silo with all the participants.

For a real-world implementation, it is crucial to take into account the frequency of communication rounds between the clients and the central server. Depending on the unique requirements and threat landscape at each silo, the update intervals may differ. Some silos might necessitate more frequent updates, perhaps on a daily, weekly, or monthly basis, to guarantee timely adaptation to emerging threats. This adaptive scheduling of communication rounds can be customized to suit the specific needs of individual silos, enabling a finely tuned response to evolving security challenges.

5.3 Evaluating the Resilience of FedNIDS against Adversarial Attacks

The next experiment is designed to provide a comprehensive assessment of the resilience of *FedNIDS* against adversarial samples at two critical stages: immediately after the initial training (Stage 1) and following fine-tuning with additional adversarial data (Stage 2). The results from this experiment offer crucial insights into how *FedNIDS* adapts to adversarial data and whether the fine-tuning process, aided by an increased quantity of adversarial samples, enhances its effectiveness.

The evaluation process was conducted as follows: We employed adversarial datasets generated by the two toolchains, as elaborated in the previous section (Section 4), for this assessment. These datasets were created to mimic potential real-world evasion attacks. During Stage 2, we divided the total number of samples that successfully evaded the DNN model using these toolchains into testing and training sets. In Stage 1, we assessed the *FedNIDS* global model using the 20% testing set of adversarial samples, which was unique to each silo. In Stage 2, the *FedNIDS* model underwent a fine-tuning process using the additional 80% of adversarial training data, which was combined with the original training samples at each silo. Subsequently, we evaluated the proposed algorithm again on the same 20% testing set of adversarial samples.

The results, as illustrated in Figure 4, demonstrate that during Stage 1, the *FedNIDS* model encounters challenges in classifying adversarial samples, resulting in an average F1 score of 0.17 across all silos for both toolchains. This difficulty can be attributed to the inherent complexity of identifying adversarial evasion samples within the federated learning global model. However, after the fine-tuning process in Stage 2, *FedNIDS* demonstrates a noteworthy improvement in accuracy against adversarial samples. The average F1 score across all silos increases to 0.92. The fine-tuning mechanism enables the model to adapt and refine its decision boundaries based on the additional adversarial data, resulting in a significant improvement (around 75%) in the F1 score. This process enhances the model's ability to learn more robust features and decision rules, ultimately enabling it to better discriminate between benign and adversarial samples. These findings hold significant implications for real-world deployment scenarios where NIDS frequently faces adversarial attempts. These experiments underscore the effectiveness of our *FedNIDS* methodology in providing an adaptive and robust defense against novel network threats while maintaining data privacy and security.

6 Conclusions and Future Works

In this study, we have presented the *FedNIDS*, a novel framework that leverages the power of federated learning and DNNs to enhance the accuracy, robustness, and privacy preservation of NIDS. Our study is the first to address the issue of non-IID data across different clients for packet-based NIDS in a federated learning framework. The framework addresses the challenge of accurate attack detection while respecting the privacy constraints of individual clients. The *FedNIDS* framework comprises two stages: federated DNN pre-training and federated novel attack fine-tuning. Through a comprehensive experimental evaluation, we demonstrated the effectiveness of *FedNIDS* in detecting specific attacks across different silos and its ability to adapt to new and emerging attack patterns. The quick adaptability of *FedNIDS* to changing attack patterns ensures that it remains effective even

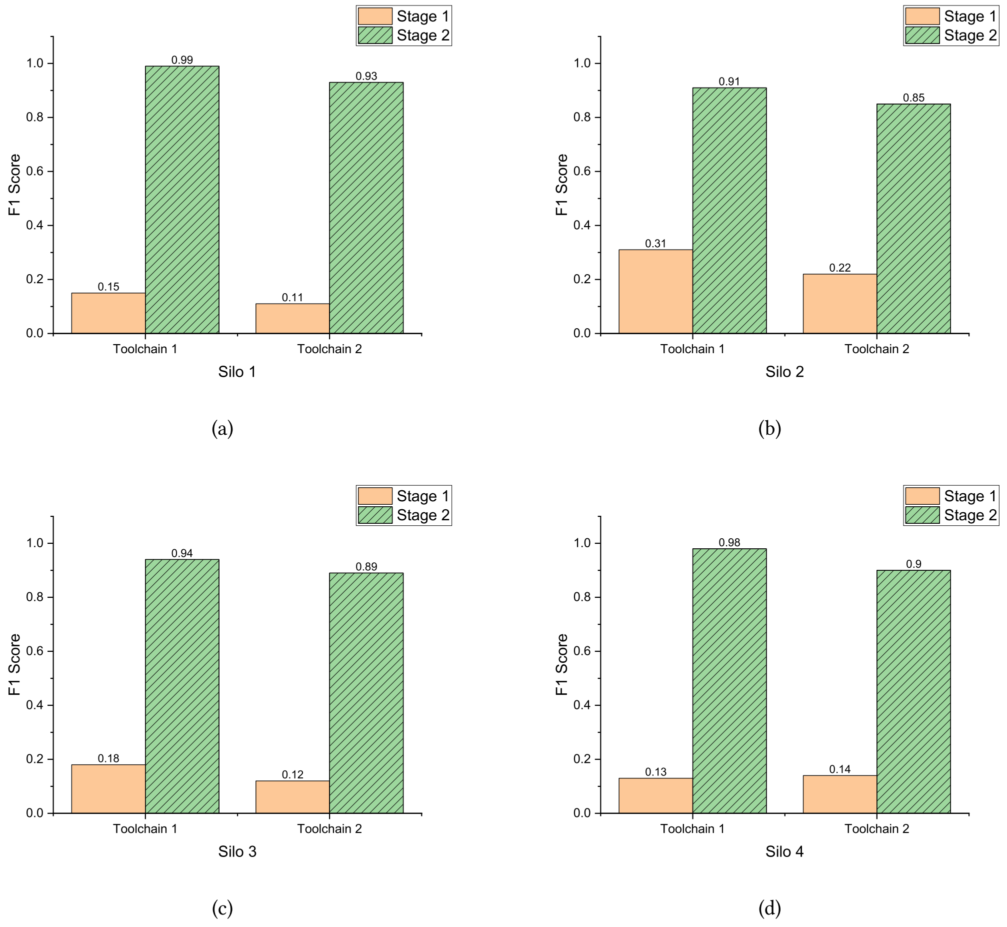


Fig. 4. The performance of FedNIDS against adversarial attacks after fine-tuning.

as new threats emerge. The robustness of our approach against adversarial attacks signifies its suitability for real-world scenarios where adversaries may attempt to evade detection.

Although our study has yielded promising results, below we discuss the assumptions in this study and practical considerations of deploying such a framework:

(1) *Reliable communication*: In our experiments, we assume reliable communication channels between the silos and the central server. In real-world scenarios, factors like latency, bandwidth constraints, or communication costs could pose practical challenges.
(2) *Privacy and security*: Our federated learning approach assumes a strong foundation of privacy and security to ensure that the learned weights transferred out from silos are secured. This is an underlying assumption in our approach and should be appropriately designed in real-world deployments.
(3) *Non-IID data in silos*: While our silo setup simulates a decentralized network, real-world conditions may introduce greater complexity in data distribution. Factors such as diverse hardware, software configurations, and network traffic patterns could intensify non-IID effects. Employing quantitative methods to assess the non-IID nature of the data within silos, such as data augmentation, transfer learning, and domain

adaptation algorithms or advanced statistical analyses, would enhance the robustness of such an approach in a real-world setting.

(4) *Novelty detection and labeling*: Our work assumes the availability of novel attack samples during the fine-tuning stage. However, in the real world, security analysts at the affected silo are tasked with identifying novel attacks that bypass the local model. These newly labeled samples are then used for the fine-tuning stage to further update the global model.

(5) *Dataset availability and proportion of attack samples*: The availability of labeled network data for security purposes is limited, which can pose challenges for training intrusion detection models. Collaboration with industry partners and network security practitioners is crucial to obtain labeled network data from real-world environments. The definition of "malicious" traffic can also be nuanced and context-dependent, and the labeling of network activities as malicious or benign may vary depending on the specific security policies and threat models of an organization. The proportion of malicious packets in our experimental setup obtained from publicly available datasets may not accurately reflect the realistic ratios encountered in live network environments, where malicious traffic often constitutes a small fraction of the overall network traffic. The network traffic volume used in our experiments, measured in gigabytes, may not fully represent the scale and complexity of heavily utilized live networks. To address these concerns, adaptive labeling and context-aware detection techniques should be developed to enhance the practicality of deploying such a system in different organizational contexts. Strategies to handle class imbalance between benign and malicious traffic in the training data should also be explored to improve the model's performance in real-world scenarios.

While our study focuses on packet-based analysis, future research could explore the integration of flow-based and packet-based approaches to further enhance network security analysis, leveraging the strengths of both techniques. One interesting avenue would be to investigate a packet-level graph-based representation. This representation would incorporate flow and application metadata, offering the intrusion detector additional context. Another avenue is examining alternative DNN architectures suitable for resource-constrained devices like IoT devices and sensors, which grapple with constraints in processing power, memory, storage capacity, and unreliable network connectivity. Furthermore, it would be valuable for the cybersecurity research community to investigate the delicate balance between accuracy and efficiency in these scenarios. Another intriguing avenue is exploring the integration of multi-modal data sources to achieve a more comprehensive understanding of network behaviors.

References

[1] UNB. 2023. Canadian Institute of Cybersecurity (CIC). Retrieved May 01, 2023 from https://www.unb.ca/cic/datasets/ids-2017.html

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

[3] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. 2022. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications* 195 (2022), 346–361.

[4] Muhammad Ahmad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. 2021. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. *EURASIP Journal on Wireless Communications and Networking* 2021, 1 (2021), 1–23.

[5] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60 (2016), 19–31.

[6] Ammar Alazab, Ansam Khraisat, Sarabjot Singh, and Tony Jan. 2023. Enhancing privacy-preserving intrusion detection through federated learning. *Electronics* 12, 16 (2023), 3382.

[7] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, and Kamal Singh. 2022. FLUIDS: Federated learning with semi-supervised approach for Intrusion Detection System. In *Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC '22).* IEEE, 523–524.

[8] Manjula C. Belavagi and Balachandra Muniyal. 2016. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science* 89 (2016), 117–123.

[9] Aitor Belenguer, Jose A. Pascual, and Javier Navaridas. 2023. GöwFed: A novel federated network intrusion detection system. *Journal of Network and Computer Applications* 217 (2023), 103653.

[10] Vance W. Berger and YanYan Zhou. 2014. Kolmogorov–Smirnov test: Overview. *Wiley Statsref: Statistics Reference Online*. Wiley Online Library

[11] David A. Bierbrauer, Michael J. De Lucia, Krishna Reddy, Paul Maxwell, and Nathaniel D. Bastian. 2023. Transfer learning for raw network traffic detection. *Expert Systems with Applications* 211 (2023), 118641.

[12] Philippe Biondi. 2010. Scapy documentation (!). Vol. 469 (2010), 155–203.

[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.

[14] Jie Cui, Hu Sun, Hong Zhong, Jing Zhang, Lu Wei, Irina Bolodurina, and Debiao He. 2023. Collaborative intrusion detection system for SDVN: A fairness federated deep learning approach. *IEEE Transactions on Parallel and Distributed Systems* 34, 9 (2023), 2512–2528.

[15] Michael J. De Lucia, Paul E. Maxwell, Nathaniel D. Bastian, Ananthram Swami, Brian Jalaian, and Nandi Leslie. 2021. Machine learning raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, Vol. 11746. SPIE, 185–194.

[16] Felix Erlacher and Falko Dressler. 2018. FIXIDS: A high-speed signature-based flow intrusion detection system. In *Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium.* IEEE, 1–8.

[17] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* 28, 1–2 (2009), 18–28.

[18] Jalal Ghadermazi, Ankit Shah, and Nathaniel Bastian. 2023. Towards real-time network intrusion detection with image-based sequential packets representation. TechRxiv Preprint. DOI: https://doi.org/10.36227/techrxiv.23291588.v1

[19] John C. Gower. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27 (1971), 857–871.

[20] Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. 2023. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 25, 1 (2023), 538–566.

[21] Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. 2020. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access* 8 (2020), 104650–104675.

[22] Vanlalruata Hnamte and Jamal Hussain. 2023. Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach. *Telematics and Informatics Reports* 11 (2023), 100077. DOI: https://doi.org/10.1016/j.teler.2023.100077

[23] Soumyadeep Hore, Jalal Ghadermazi, Diwas Paudel, Ankit Shah, Tapas K. Das, and Nathaniel D. Bastian. 2023. Deep PackGen: A deep reinforcement learning framework for adversarial network packet generation. arXiv:2305.11039. Retrieved from https://doi.org/10.48550/arXiv.2305.11039

[24] Soumyadeep Hore, Quoc Nguyen, Yulun Xu, Ankit Shah, Nathaniel Bastian, and Trung Le. 2023. Empirical evaluation of autoencoder models for anomaly detection in packet-based NIDS. In *Proceedings of the 2023 6th IEEE Conference on Dependable and Secure Computing*. IEEE, 1–8.

[25] Meryem Janati Idrissi, Hamza Alami, Abdelkader El Mahdaouy, Abdellah El Mekki, Soufiane Oualil, Zakaria Yartaoui, and Ismail Berrada. 2023. Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems. *Expert Systems with Applications* 234 (2023), 121000.

[26] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi. Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[27] Min-Joo Kang and Je-Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one* 11, 6 (2016), e0155781.

[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*. Hal Daumé III and Aarti Singh (Eds.), Proceedings of Machine Learning Research, Vol. 119, PMLR, 5132–5143. DOI: https://proceedings.mlr.press/v119/karimireddy20a.html

[29] Christopher Krügel, Thomas Toth, and Engin Kirda. 2002. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 201–208.

[30] Vinod Kumar and Om Prakash Sangwan. 2012. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology* 1, 3 (2012), 35–41.

[31] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. 2020. DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics* 17, 8 (2020), 5615–5624.

[32] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 3347–3366.

[33] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[34] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.

[35] Patrick E. McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini Encyclopedia of Psychology*, 1–1.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics.* PMLR, 1273–1282.

[37] Mohamed Amine Merzouk, Frédéric Cuppens, Nora Boulahia-Cuppens, and Reda Yaich. 2023. Parameterizing poisoning attacks in federated learning-based intrusion detection. In *Proceedings of the 18th International Conference on Availability, Reliability and* Security, 1–8.

[38] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. 2002. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN '02) (Cat. No. 02CH37290)*, Vol. 2, IEEE, 1702–1707.

[39] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS '19).* IEEE, 756–767.

[40] Guansong Pang, Chunhua Shen, and Anton van den Hengel. 2019. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 353–362.

[41] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. 2020. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network* 34, 6 (2020), 310–317.

[42] Md Mamunur Rashid, Shahriar Usman Khan, Fariha Eusufzai, Md Azharuddin Redwan, Saifur Rahman Sabuj, and Mahmoud Elsharief. 2023. A federated learning-based approach for improving intrusion detection in industrial internet of things networks. *Network* 3, 1 (2023), 158–179.

[43] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H. Brendan McMahan. 2020. Adaptive federated optimization. arXiv:2003.00295. Retrieved from https://doi.org/10.48550/arXiv.2003.00295

[44] Pedro Ruzafa-Alcázar, Pablo Fernández-Saura, Enrique Mármol-Campos, Aurora González-Vidal, José L. Hernández-Ramos, Jorge Bernal-Bernabe, and Antonio F. Skarmeta. 2021. Intrusion detection based on privacy-preserving federated learning for the industrial IoT. *IEEE Transactions on Industrial Informatics* 19, 2 (2021), 1145–1154.

[45] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2019. A detailed analysis of the cicids2017 data set. In *Information Systems Security and Privacy: 4th International Conference (ICISSP '18).* Springer, 172–188.

[46] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization (*ICISSp '18)* 1 (2018), 108–116.

[47] Dug Song and Contributors. 2009. dpkt 1.9.2 documentation. Retrieved February 16, 2023 from https://dpkt.readthedocs.io/en/latest/

[48] K. Muthamil Sudar and P. Deepalakshmi. 2022. Flow-based detection and mitigation of low-rate DDOS attack in sdn environment using machine learning techniques. In *IoT and Analytics for Sensor Networks: Proceedings of ICWSNUCA 2021.* Springer, 193–205.

[49] Pu Tian, Zheyi Chen, Wei Yu, and Weixian Liao. 2021. Towards asynchronous federated learning based threat detection: A DC-Adam approach. *Computers & Security* 108 (2021), 102344.

[50] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. 2017. Flow-based intrusion detection: Techniques and challenges. *Computers & Security* 70 (2017), 238–254.

[51] Nguyen Thanh Van, Tran Ngoc Thinh, and Le Thanh Sach. 2017. An anomaly-based network intrusion detection system using deep learning. In *Proceedings of the 2017 International Conference on System Science and Engineering (ICSSE '17).* IEEE, 210–214.

[52] Miel Verkerken, Laurens D'hooge, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2022. Towards model generalization for intrusion detection: Unsupervised machine learning techniques. *Journal of Network and Systems Management* 30 (2022), 1–25.

[53] Bo Wang, Yang Su, Mingshu Zhang, and Junke Nie. 2020. A deep hierarchical network for packet-level malicious traffic detection. *IEEE Access* 8 (2020), 201728–201740.

[54] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. arXiv:2002.06440. Retrieved from https://doi.org/10.48550/arXiv.2002.06440

[55] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems* 33 (2020), 7611–7623.

[56] Mikal R. Willeke, David A. Bierbrauer, and Nathaniel D. Bastian. 2023. Data-efficient, federated learning for raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, Vol. 12538, SPIE, 247–262.

[57] Kai Yang, Yuanming Shi, Yong Zhou, Zhanpeng Yang, Liqun Fu, and Wei Chen. 2020. Federated machine learning for intelligent IoT via reconfigurable intelligent surface. *IEEE Network* 34, 5 (2020), 16–22.

[58] Run Yang, Hui He, Yulong Wang, Yue Qu, and Weizhe Zhang. 2023. Dependable federated learning for IoT intrusion detection against poisoning attacks. *Computers & Security* 132 (2023), 103381.

[59] Binhang Yuan, Song Ge, and Wenhui Xing. 2020. A federated learning framework for healthcare IoT devices. arXiv:2005.05083. Retrieved from https://doi.org/10.48550/arXiv.2005.05083

[60] Si-si Zhang, Jian-wei Liu, and Xin Zuo. 2021. Adaptive online incremental learning for evolving data streams. *Applied Soft Computing* 105 (2021), 107255.

Received 10 November 2023; revised 26 July 2024; accepted 28 August 2024

On Collaboration and Automation in the Context of Threat Detection and Response with Privacy-Preserving Features

LASSE NITZ and MEHDI AKBARI GURABI, Fraunhofer FIT, Sankt Augustin, Germany and RWTH Aachen University, Aachen, Germany
MILAN CERMAK, Masaryk University, Brno, Czech Republic
MARTIN ZADNIK, CESNET, Prague, Czech Republic
DAVID KARPUK, W/Intelligence, WithSecure Corporation, Helsinki, Finland
ARTHUR DRICHEL, SEBASTIAN SCHÄFER, and BENEDIKT HOLMES, RWTH Aachen University, Aachen, Germany
AVIKARSHA MANDAL, Fraunhofer FIT, Sankt Augustin, Germany

Organizations and their security operation centers often struggle to detect and respond effectively to an extensive quantity of ever-evolving cyberattacks. While collaboration, such as threat intelligence sharing between security teams, and response automation are often discussed in the cybersecurity community, issues like data sensitivity and confidence in detection may hinder their adoption. This work investigates the potentials and challenges of collaboration and automation to enhance incident response processes. We propose a reference architecture for data sharing in threat detection and response, aiming to boost collaborative and automated efforts across organizations while also considering privacy-preserving features. To address these challenges and potentials, we discuss how such a framework could enhance current response processes within and between organizations, validated with results in local attack detection, incident response, and data sharing.

CCS Concepts: • **Security and privacy** → **Malware and its mitigation**; **Intrusion detection systems**; *Data anonymization and sanitization*; *Privacy-preserving protocols*; • **Information systems** → **Collaborative and social computing systems and tools**; • **Computing methodologies** → *Machine learning*; • **Human-centered computing** → *Visualization*;

Additional Key Words and Phrases: Cybersecurity, collaborative detection and response, incident response automation, information sharing, privacy

This work has received funding from the EU H2020 LEIT Information and Communication Technologies programme for the project SAPPAN (Sharing and Automation for Privacy Preserving Attack Neutralization) under grant agreement No. 833418.
Authors' Contact Information: Lasse Nitz (corresponding author), Fraunhofer FIT, Sankt Augustin, Germany and RWTH Aachen University, Aachen, Germany; e-mail: lasse.nitz@fit.fraunhofer.de; Mehdi Akbari Gurabi, Fraunhofer FIT, Sankt Augustin, Germany and RWTH Aachen University, Aachen, Germany; e-mail: mehdi.akbari.gurabi@fit.fraunhofer.de; Milan Cermak, Masaryk University, Brno, Czech Republic; e-mail: cermak@ics.muni.cz; Martin Zadnik, CESNET, Prague, Czech Republic; e-mail: zadnik@cesnet.cz; David Karpuk, W/Intelligence, WithSecure Corporation, Helsinki, Finland; e-mail: davekarpuk@gmail.com; Arthur Drichel, RWTH Aachen University, Aachen, Germany; e-mail: drichel@itsec.rwth-aachen.de; Sebastian Schäfer, RWTH Aachen University, Aachen, Germany; e-mail: schaefer@itsec.rwth-aachen.de; Benedikt Holmes, RWTH Aachen University, Aachen, Germany; e-mail: holmes@itsec.rwth-aachen.de; Avikarsha Mandal, Fraunhofer FIT, Sankt Augustin, Germany; e-mail: avikarsha.mandal@fit.fraunhofer.de.

ACM 2576-5337/2025/2-ART5
https://doi.org/10.1145/3707651

ACM Reference format:
Lasse Nitz, Mehdi Akbari Gurabi, Milan Cermak, Martin Zadnik, David Karpuk, Arthur Drichel, Sebastian Schäfer, Benedikt Holmes, and Avikarsha Mandal. 2025. On Collaboration and Automation in the Context of Threat Detection and Response with Privacy-Preserving Features. *Digit. Threat. Res. Pract.* 6, 1, Article 5 (February 2025), 36 pages.
https://doi.org/10.1145/3707651

1 Introduction

In the past few years, there has been a significant rise in cyberattacks across various sectors impacting our society and economy. According to the **European Union (EU)** Agency for Network and Information Security Threat Landscape Report 2023 [38], there has been a more than 150% increase in globally reported incidents from the last half of 2022 to the first half of 2023. Ransomware and malware threats are on the rise, where cybercriminals are effectively utilizing ransomware/malware-as-a-service and purchasing sophisticated malware from underground sources to launch targeted attacks [38]. Among the ongoing political tensions in regions like Russia-Ukraine and Israel-Palestine, threat actors are targeting critical infrastructures such as the energy and defense sector with malware and social engineering-based attacks [65].

This rapidly evolving threat landscape presents a substantial challenge for many organizations. Cybersecurity incidents, also caused by advanced persistent threats, are omnipresent and call for rapid and effective detection and response [9, 26]. Hence, organizations must employ effective threat detection and response processes starting from understanding their technical and organizational environment to appropriately responding to cyberattacks and recovering from their consequences [87]. Moreover, small- or mid-sized enterprises do not have sufficient resources or expertise to face advanced threats [20]. Thus, they need to rely on outsourcing threat detection and response to external cybersecurity service providers. In such cases, the privacy of user data and the confidentiality of an organization's business information become major concerns while outsourcing.

Additionally, **Security Operations Centers (SOCs)**, Computer Emergency Response Teams, and **Computer Security Incident Response Teams (CSIRTs)** are grappling with a significant shortage of skilled personnel [96]. SOC analysts often face burnout while manually filtering out a large number of false-positive alerts. Given the sheer volume of false-positive alerts, alerts caused by actually malicious activities may go unnoticed and undetected amid background noise, thereby increasing the risk of damage to organizational assets [7]. In intrusion detection, rule-based and signature-based approaches suffer from identifying new types of attacks with unseen patterns, whereas, **Machine Learning (ML)**-based approaches tend to excel at identifying new types of threats, and detecting anomalies. However, the lack of interpretability in ML models may hinder their usage, as analysts may find decisions from rule-based approaches more understandable while investigating false alarms [46]. In the case of correctly triggered incidents, SOC operators manually initiate specific response actions based on factors such as the type of detected attack, the confidence level linked to the detection, the significance of the asset, and the evaluation of the incident's severity. The response actions can be quite repetitive and performing them manually may lengthen the response time. Therefore, recommending suitable response actions or even automatically executing common response actions for certain types of attacks based on their severity could significantly enhance the response workflow in SOCs.

While addressing the aforementioned challenges, SOC teams may immensely benefit from collectively and collaboratively sharing **Cyber Threat Intelligence (CTI)** [18, 19]. CTI encompasses a wide range of information, whether tactical, strategic, or operational, designed to detect and handle incidents [60]. Recent work from Preuveneers and Joosen [82] pointed out that CTI such as traditional **Indicators of Compromise (IoCs)** could lose its impact over time due to its volatile nature (e.g., the IP address of a compromised machine may change) and proposed an access control-based sharing scheme of ML models for threat detection. Sharing ML-based detection models as the new-generation IoCs has the potential to help organizations detect and respond to new types of attacks on time.

Subsequently, privacy issues need to be handled carefully before or during the sharing of CTI between organizations, especially since CTI may include personal information about end users, confidential company information, and sensitive infrastructure details. Organizations could collaboratively train threat detection models on incident data by utilizing **Privacy-Enhancing Technologies (PETs)** such as federated learning, data sanitization for pre-processing, or cryptographic techniques. Especially federated ML techniques can facilitate collaborative attack detection between organizations and could improve the incident detection accuracy by using shared global ML models, e.g., for the purpose of anomaly or malware detection. By collaboration, one could not only reduce biases in the training set but may also achieve higher accuracy and reduced false positives. Organizations could also benefit from sharing cybersecurity playbooks [84], which provide step-by-step guidelines for response to and recovery from different types of incidents. However, cybersecurity playbooks are often organization-specific and are documented in unstructured or semi-structured formats [3]. Thus, there is a demand for standardized, machine-readable, interoperable playbook formats to facilitate sharing and seamless integration into the local system, which could enable further automation.

Improving an organization's incident response capabilities and collaborations are also important from a legal perspective, not only due to privacy laws [8] but also with emerging cybersecurity regulations. For example, in early 2023, the EU approved a new version of the **Network and Information Systems Directive 2 (NIS2)** [86]. NIS2 defines the EU's cybersecurity priorities for enhanced security and resilience in organizations against evolving threats. To comply with such legislation, companies must prepare themselves to meet several cybersecurity requirements such as following certain incident response procedures (e.g., prompt incident reporting to national authorities). In case of non-compliance, companies may have to pay hefty fines.

Our Contributions. In this article, we contribute to various aspects of threat detection and incident response, which can be presented fourfold. First, we identify key potentials and challenges of privacy-preserving collaboration and automation in threat detection and incident response. Our insights are derived from a practical perspective, based on our experiences gained in the EU H2020 cybersecurity project named **Sharing and Automation for Privacy-Preserving Attack Neutralization (SAPPAN)**.[1] The project comprised of a consortium with 77 individuals from 8 organizations, involving security providers, CSIRTs, corporate entities, and academic partners. Second, we propose a reference architecture to enable CTI sharing between organizations. The architecture includes different technical components and their interactions to facilitate privacy-preserving sharing of CTI and response automation in a generic organization that wants to participate in collaboration. Third, we present an extensive array of results in threat detection and response, clustered into three tasks: local attack detection, incident response, and sharing. The local attack detection task surveys our ML-based approaches to improve malware detection and anomaly detection. The incident response task contributes to three topics: cybersecurity playbook management, incident similarity models for recommendation and support of human operators, and incident response automation by demonstrating an automated malware analysis workflow. In the sharing task, we contribute to privacy-preserving collaboration such as federated approaches for threat detection, data sanitization approaches for sharing, and cybersecurity playbook sharing. Fourth, drawing from our validation results, we retrospectively discuss the efficacy and limitations of different results by mapping them to relevant challenges and potentials, and aligning them with the proposed reference architecture. We further outline directions for future research.

Outline. The remainder of this work is structured as follows: Section 2 provides essential background on the incident response lifecycle. Section 3 presents our identified list of key potentials and challenges of collaboration and automation when adapted to the incident response lifecycle. Section 4 proposes our framework and presents a reference architecture for a general incident detection and response system, covering a description of individual technical components and their interactions. Sections 5–7 cover our approaches and developed components for the tasks of local attack detection, incident response, and sharing, respectively. Section 8 discusses our evaluation

[1]SAPPAN on CORDIS: https://doi.org/10.3030/833418; SAPPAN project Web site: https://sappan-project.eu/

results for each task, validated with technically quantifiable metrics as well as opinion-based perspectives. Section 9 retroactively discusses our results from the perspectives of challenges and potentials, as well as the proposed reference architecture. Lastly, Section 10 concludes this work and provides directions for future research.

2 Background

Organizations vary in the maturity of their detection and response capabilities, and they also interpret incidents differently, thus making it difficult to establish unified response strategies [26]. An incident response process model can help organizations to plan their response strategies in a standardized way. There exist several process models for incident responses. Two of the prominent ones are the **National Institute of Standards and Technology (NIST)** incident response lifecycle [25], and the SANS Institute's PICERL model [74]. Most of the frameworks present similar stages of response with slight variations in wording and details. To understand the common response stages, we briefly discuss the NIST incident response lifecycle, which consists of four main phases [25].

The *Preparation* phase includes building up the IT infrastructure, software, and hardware for the response team, as well as ensuring the functionality of monitoring tools and reporting mechanisms. During this phase, organizations aim to minimize incidents by implementing controls such as risk assessment and malware protection. Additionally, organizations increase awareness and provide training to their analysts to improve incident handling.

In the *Detection and Analysis* phase, incident data is collected to detect past/ongoing incidents (indicators) or potential future incidents (precursors). Sources for detecting indicators or precursors include **Intrusion Detection/Prevention Systems (IDS/IPS)**, antivirus and antispam software, application and network logs, and publicly available information such as the National Vulnerability Database. While most IDS/IPS products, commonly utilizing attack signatures or statistical detection methods to identify unusual activities, produce alerts with relevant data, they often generate false positives, requiring manual analysis and validation of alerts. Moreover, incidents are assessed and prioritized based on their impact. There are also obligations to notify affected entities according to policies and legal frameworks (e.g., NIS2 mandates an initial report to be submitted within 24 hours [86]).

During the *Containment, Eradication, and Recovery* phase, the core incident response activities take place. The containment part includes essential decision-making (e.g., whether to initiate a system shutdown or to isolate parts of a network) based on pre-defined strategies or playbooks, depending on acceptable risks to avoid further damage. Forensic techniques are also employed to gather evidence, simultaneously documenting logs for resolving the issue and meeting legal requirements. When the incident is contained, analysts focus on eradication and recovery, aiming to restore system operations to a normal state. This may involve tasks such as restoring backups, rebuilding systems, applying patches, and increasing security measures to prevent similar incidents.

The *Post-Incident Activities* phase includes lessons learned, and reviews with all involved parties after major incidents to improve future incident response. This retrospective phase initiates technical measures, e.g., by evaluating whether collected incident data can be utilized in the preparation phase for purposes such as risk assessment or implementation of additional controls.

Our Approach. Figure 1 presents our collaboration approach for threat detection and response aligned with the NIST incident response framework. In the local view, different phases of the incident response lifecycle take place within an organization. In the collaborative view, CTI is shared from the local view in different phases of the lifecycle to establish a common understanding of attack detection, incident assessment, and handling. The privacy issues are handled before or during sharing of CTI. Moreover, the processes can be automated at any phase of the response lifecycle. However, in this work, we only investigate automating processes within response activities, specifically focusing on the *Containment, Eradication, and Recovery* phase, as illustrated in the local view in Figure 1.

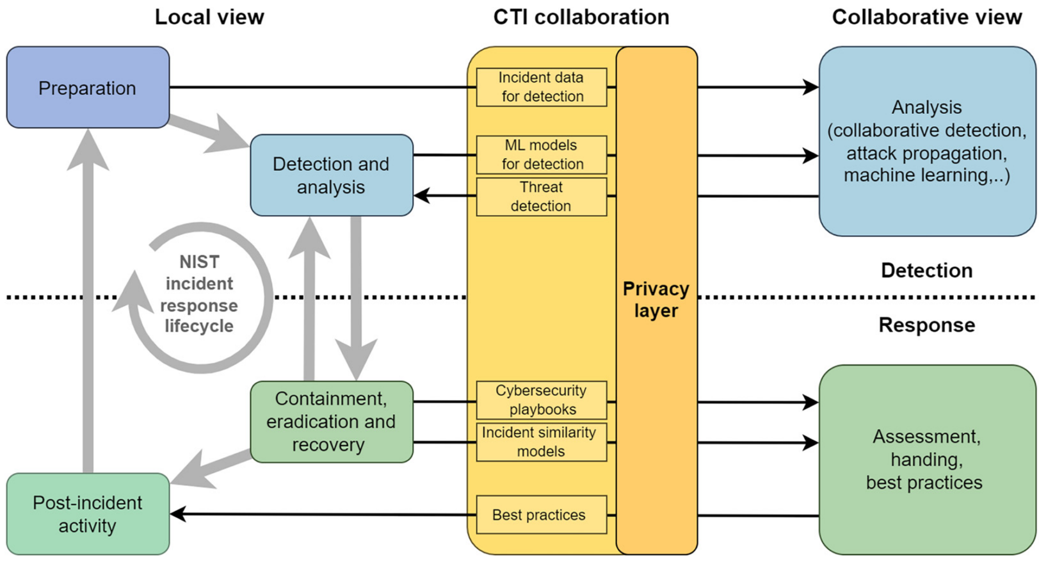

Local view
CTI collaboration
Collaborative view
Preparation
Detection and analysis
NIST incident response lifecycle
Containment, eradication and recovery
Post-incident activity
Incident data for detection
ML models for detection
Threat detection
Cybersecurity playbooks
Incident similarity models
Best practices
Privacy layer
Analysis (collaborative detection, attack propagation, machine learning,..)
Detection
Response
Assessment, handing, best practices

Fig. 1. SAPPAN collaboration concept aligned with the National Institute of Standards and Technology (NIST) incident response lifecycle.

3 Key Potentials and Challenges

To gain a better understanding of how collaboration and automation can affect work in the cybersecurity domain, we have identified key potentials that they may bring as well as key challenges that need to be overcome by respective solutions. For this, we decided to take a 2-step divergence-convergence approach, which first creatively explores different options and then analytically condenses the initial findings down to a refined result. This reflects the problem identification phase (consisting of a discovery and a definition step) of the double diamond model, a popular process model in design thinking (see, e.g., [73]). A total of 17 experts of the SAPPAN consortium participated directly in the identification of key potentials and challenges, which allowed to capture a great variety of different perspectives from both industry and academia. The experts' backgrounds cover the areas of threat detection, incident handling, security services and consultation, ML and data science, PETs, and data visualization in the cybersecurity domain.

In the first step, each participating expert has been asked to think of and document possible potentials and challenges of integrating collaboration and automation into cybersecurity use cases. This part of the process did not include active exchange between the different participants. For the second step, we have organized an interactive workshop with the goal of condensing the collected potentials and challenges down to the key ones. To achieve this, the participants have first been tasked with identifying clusters in the collected potentials and challenges. Subsequently, the clusters have been discussed. For this part of the process, the goal was to summarize the points within a cluster, and to define a generalized description for each cluster. Afterwards, the cluster descriptions were checked against each other to identify possible overlaps or conflicts.

This process has been carried out separately for collaboration and automation. The resulting key potentials and challenges are listed in Table 1 for collaboration and Table 2 for automation. These key potentials and challenges are further used as guidance for the discussion of the validation results as given in Section 9.

While our identification of challenges and potentials primarily focuses on technical aspects, recent work has also looked at challenges and potentials from the perspective of interrelationships of technical and social aspects in CTI sharing communities [45]. As such, our work complements these findings from a technical point of view.

Table 1. Challenges and Potentials in Collaboration

ID	**Name**	**Description**
Collaboration challenges		
C.C1	Sensitive information	Concerns about leaking sensitive information include customer privacy violations resulting in reputation damage, organizational vulnerabilities exposed to adversaries, and regulatory non-compliance.
C.C2	Machine-interpretable data	Machine-interpretable data is essential for enabling effective collaboration on a larger scale. It is, for example, important for finding relevant data, assessing the quality of shared data, and using shared data in a plug-and-play manner.
C.C3	Data model universality	The data model and its format must be widely accepted to enable the exchange of data between a large variety of actors.
C.C4	Data model adaptability	The data model must be flexible enough to accommodate future extensions such as different types of IoCs, attack types, or response information.
C.C5	Quality assurance	The quality of exchanged information determines if a collaboration is beneficial or not. It includes multiple aspects such as data relevance, timeliness, completeness, noisy and false information.
Collaboration potentials		
C.P1	Preemptive defense	Preparation against novel kinds of attacks, even before they were observed in the own organization by sharing data identifying the attack or the attack source (learning from others' experiences).
C.P2	Collaborative data generation and learning	Collaborative creation of balanced datasets and collaborative learning. The supervised ML-based detection of malicious behavior requires balanced and diversified datasets to reduce false positives, which may benefit from sharing information about the availability of datasets and their rating, or collaborative/federated ML operating over the sharing platform.
C.P3	Standardization	Standardized description to enable machine understanding as well as wide deployment.
C.P4	Validation via visualization	Increased trust in data/models through interpretability, even when considering large amounts of data, complex problem domains, or privacy concerns.

4 The SAPPAN Reference Architecture

As the key potentials and challenges identified in the previous section partly result from interactions of different technical components of a sharing-enabled incident response system, it is important to understand the context in which these components interact. For this purpose, we propose a general architecture, which we describe further below and refer to as the SAPPAN reference architecture. A visual representation of this architecture is given in Figure 2.

To align the different components of the SAPPAN reference architecture with different purposes as well as different stages in the incident response life cycle, we cluster these components into three tasks: local attack detection, incident response, and sharing. In the following, we briefly introduce the purpose of each component of the architecture in accordance with this clustering. We further utilize the clustering into these three tasks in subsequent sections to discuss the background and our contributions to each of them. The SAPPAN reference architecture is also revisited in Section 9 to discuss the implications of our validation results concerning the architecture and its components.

Table 2. Challenges and Potentials in Automation

ID	Name	Description
Automation challenges		
A.C1	Risk of disruption	The risk of disruptions, where ill-defined procedures might cause damages.
A.C2	Implementation effort	Implementing automation demands weighing costs against benefits. Identifying appropriate scenarios and integrating them with existing systems is essential.
A.C3	Nuances of incidents	Incidents' nuances and their complex nature pose challenges, particularly in decision-making for branching within automated workflows as we transition from general to automated playbooks.
A.C4	Confidence in automated decisions	Confidence in automation decisions and ensuring good data quality is foundational for the success of the automated response.
Automation potentials		
A.P1	Workload reduction	Automation has the potential to substantially reduce the workload of human operators. Handling routine incidents might allow human operators to focus on more complex/unusual incidents and mitigate alert fatigue.
A.P2	Faster processing and resolution of incidents	Timely incident processing and resolution are potential benefits of automation. Associated potential benefits are its scalability, non-reliance on working hours, and trimmed operational costs.
A.P3	Integrated logging	Integrated logging in automation ensures every action is tracked, simplifying statistics generation and negating potential employee surveillance concerns. If no humans are involved, employee rights do not have to be considered, which might otherwise be a concern.
A.P4	Increased quality via human–machine teaming	Human–machine teaming carries the potential to augment the quality of ML models. Visual analytics can further work as a gap-filler for non-automatable parts in response and recovery and can be used for monitoring the system as well as recommendation and validation purposes.

4.1 Local Attack Detection

The *system and network monitoring* component takes care of all activities related to observing ongoing activities. Its primary purpose within the SAPPAN reference architecture is to collect data, which is generated as part of system monitoring. This data is then shared with components that require such data as input, e.g., to create detection system components or to detect malicious activities within live traffic.

The *ML* component is intended to train models and coordinate activities related to collaborative efforts in ML, such as federated learning. It provides trained models for the detection system and the ML visualization component.

The component for *ML visualization* aims to support human users to better understand the trained models by providing suitable visualizations.

For the purpose of detecting incidents in the monitored system, respective monitoring data is provided to the *detection and analysis system*. In this context, trained classifiers can be utilized to detect malicious samples or patterns in the data provided to the detection and analysis system.

Monitoring data is further provided to a *profiling* component, which creates profiles based on traffic observed in the past. The purpose of these profiles is to capture the usual behavior of monitored system actors (such as hosts or applications), which can further be utilized for detecting anomalous behavior on new monitoring data.

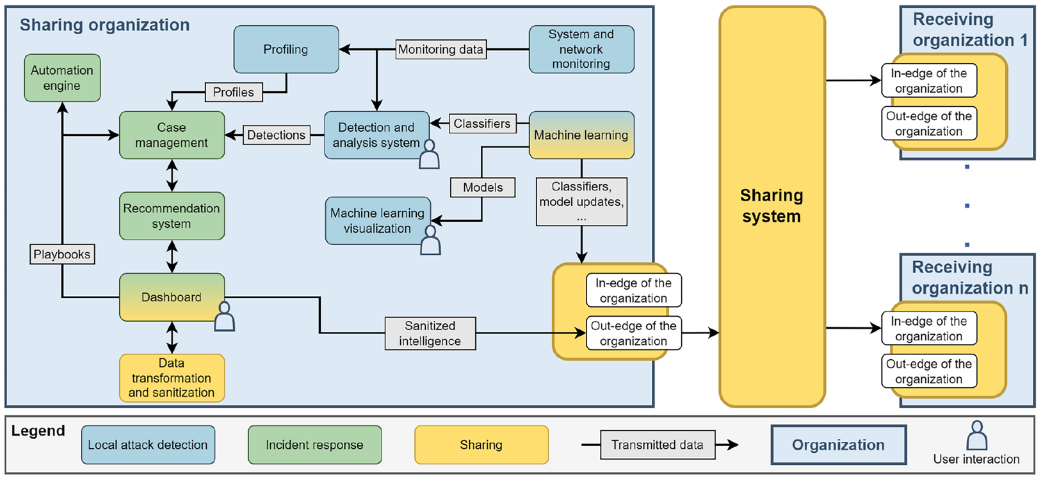


Fig. 2. Graphical representation of the SAPPAN reference architecture. Different technical components are represented as nodes and interactions between them as edges. Components with user interactions are marked respectively. Note that this figure depicts the sharing flow of an organization. When receiving data, some of the edges change direction.

4.2 Incident Response

The *case management* component uses profiles and detections by the respective components to organize activities related to new incidents, actions taken, and the archiving of old incidents.

The *recommendation system* analyzes either how specific cases have been handled in the past to recommend actions for similar cases to a human operator or implements its own heuristics to infer the next actions.

The *dashboard* aims to guide a human user through the incident handling process and further provides the interface to communicate with the sharing system. The dashboard is a user interface to access and interact with the systems to benefit from past cases through recommendations. Actions taken by the operator can further be utilized for future recommendations.

In order to reduce the manual work for human operators, the *automation engine* is intended to automatically resolve frequent low-impact incidents. This requires initial implementation efforts, in which cybersecurity playbooks are specified into executable source code, which interacts with existing systems that are deployed within the organization.

4.3 Sharing

Since data sharing introduces requirements surrounding normalization and data protection, a *data transformation and sanitization* component is connected to the dashboard. A human operator can apply sanitization and normalization operations there to comply with respective expectations and obligations before making data accessible to external parties.

Because the human operator interacts with the data transformation and sanitization component via the *dashboard*, the option to initialize data sharing after the data transformation and sanitization process is also included in the dashboard's functionality. To provide the human operator with a user interface for coordinating the interaction with the sharing system, the dashboard is connected to the in- and out-edge of the organization.

Also the *ML* component has to deal with aspects of sharing, for example, to exchange model updates with collaborators in federated learning tasks. As such processes do not require that model updates be exchanged manually by a human operator, the ML component is directly connected to the sharing system via the organization's in- and out-edges.

While not necessarily part of any of a sharing/receiving organization's infrastructure, the *sharing system* connects the different organizations and ensures that shared data is distributed to the intended recipients. As such, it is connected to the in- and out-edges of the participating organizations, which provide an interface between the sharing system and an organization's internal systems.

5 Task: Local Attack Detection

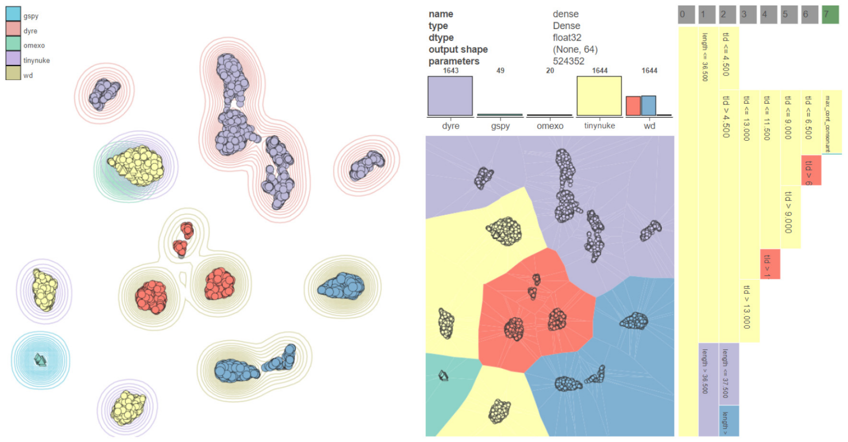
The first step in responding to an incident and ensuring the security of a network and its services is detecting and analyzing intrusions at the local network level [27]. This can be challenging for security specialists due to a large amount of diverse data and the need for the detection of various threats while reducing false positives. In the cybersecurity domain, the following three data groups based on their origin are typically used: network-based data (observed in a computer network), endpoint data (observed in computers and similar devices), and open source intelligence (cyberattack-related data that can be retrieved from other, usually publicly available, sources) [55]. Effective processing and analysis of such data then directly affect threat detection and timely mitigation capabilities. The anomaly and intrusion detection domain is a broad research area, and many different techniques and approaches emerged through the years [41]. In general, these techniques may be categorized into categories: statistical (time-series, univariable, etc.), ML, and others (signatures, threshold values, etc.) [99]. It is not possible to unambiguously determine which technique fits for what purpose, since factors such as the type of infrastructure, type of processed data, speed of data processing, the accuracy of the method (**False-Positive Rate (FPR)** and false-negative rate), or its universality, for example, need to be taken into account [21]. The key element is also an integration of these techniques into the incident response workflow and overall architecture, as described in Section 4.

Our research focused on two areas representing different analytical approaches facing challenges of local attack detection. The first is malicious domain detection, which is associated with malware and phishing attacks, representing today's main threats [57]. The second challenge we addressed is anomalous behavior detection both at the network and host levels. Our goal is to detect compromised devices even in the case of advanced, previously unknown attacks. These areas and their corresponding components are shown in Figure 2 in dark blue. The proposed methods can operate independently, but their main strength is gained when properly connected to data sharing and other incident response phases, as described in the following Task-related sections.

5.1 Malicious Domains Detection

Regarding phishing attacks, we focused on analyzing domains in **Certificate Transparency (CT)** logs, allowing us to obtain information about new domains quickly. In our research [31], we presented and evaluated a system designed to identify phishing certificates within CT logs in real time, as they are appended. In addition to real-time detection, this system supports retrospective analyses and addresses the issue of missing reference data by providing a validation mechanism. Our modular and scalable system provides a robust platform for developing new certificate detection algorithms and allows their benchmarking against current methods. Moreover, it aids in the creation of authentic datasets that are useful for training various ML classifiers.

Besides phishing attacks, we have also focused on malware-related domains, specifically domains generated by **Domain Generation Algorithms (DGAs)** used to contact **Command and Control (C2)** servers. In our related study [34], we introduced two innovative classifiers based on **Residual Neural Networks (ResNets)**. Through a comprehensive comparison with other leading classifiers, we demonstrated the effectiveness of our models. Specifically, in two binary classification scenarios (distinguishing **Algorithmically Generated Domains (AGDs)** from known DGAs and identifying AGDs from unknown DGAs), we showed that our binary classifier not only achieves top-tier classification accuracy but also has the lowest FPR. Additionally, we confirmed the real-time performance of our classifiers, proving that our ResNet-based binary classifier effectively identifies samples from previously unknown DGAs.



name	dense
type	Dense
dtype	float32
output shape	(None, 64)
parameters	524352

Fig. 3. The analysis view provides understandable interpretations of a DGA classification model [15].

To properly handle incident response, it is necessary to use existing detection methods and create new ones capable of responding to current threats. One relevant aspect in this context is the explainability of a newly developed model, which we have also focused on in our research [15]. We designed and tested a visual analytics system that enables designers of deep learning models for multi-class DGA classification to explore the patterns their models detect in the data. This allows users to analyze the progression of a customized subset of the dataset through the different layers of a deep learning model. We used clustering to identify patterns in the activations of the model's nodes and presented these patterns to the user with a decision tree for interpretation. An example of the analysis view is shown in Figure 3. Our evaluation of the system demonstrated how it can be used to better understand misclassifications, identify potential biases, and interpret the roles of different layers in a model.

5.2 Anomalous Behavior Detection

In general, intrusion detection methods can be divided into pattern-based and anomaly-based approaches. Pattern-based methods offer higher accuracy but may not be able to detect advanced or novel types of attacks. To address this issue, detection methods based on monitoring device behavior and identifying deviations from the normal state are used. These methods have a greater potential to identify even previously unknown types of attacks. Therefore, our research has focused on application and endpoint profiling serving as a basis for anomaly detection within the incident response context.

The goal of application profiling, or application fingerprinting, is to determine typical characteristics of an application, e.g., queried domains, as well as modeling the distinct behavior of an application in order to identify and classify the application. For application identification, we have proposed and evaluated rule-based process mining models that are able to identify whether certain applications are running on a device based on DNS traffic. This is, e.g., useful for network administrators or pentesters to get an overview of a network, but also for incident response handlers to get more context of the machine that produced the alert. For the application classification task, we have developed and successfully integrated into the SAPPAN architecture an approach to build process mining models based on system events. The idea is to build a model representing the typical behavior of an application in order to detect deviations, e.g., if the application is behaving maliciously.

Application behavior analysis also closely relates to analyzing processes and their relations aiming to detect anomalous events in endpoints. We have developed a model for identifying anomalous process launch events [56]. In order to increase the reliability of detections reported by the model and to support security analysts in handling those detections, we have experimented with combining detected anomalies in so-called provenance graphs. Further, we have developed a set of models designed to detect specific classes of anomalous endpoint behavior and a method for presenting connections among detected anomalies as a node-edge graph supporting decisions by incident investigators. We further supported their decision-making by creating an endpoint profiling tool, based on our 1-year study [53], that allows the analyst to classify hosts in the network and determine their expected behavior.

6 Task: Incident Response

Following the detection of an incident, an adequate response needs to be taken. However, the increasing volume of security data introduces challenges for organizations, as processing it requires more resources and leads to higher costs. With many security units understaffed and underfunded, alternative cybersecurity strategies are essential [1]. The growing infrastructure, including mobiles, cloud, and IoT devices, means more threats and alerts. To counteract this, security teams must be equipped with the latest threat intelligence knowledge, including context, methods, indications, and actionable advice that informs about potential cyber threats [102]. Gartner [44] defines this as essential information that helps understand current or emerging threats to IT or information assets. An organization's incident response approach addresses cyberattacks through specific policies and procedures aimed at the containment of, as well as recovery from cyberattacks. Repetitive and tedious tasks often frustrate SOC staff, making it challenging to retain skilled members familiar with organizational systems and processes [101]. Using threat intelligence platforms to compile, aggregate, and organize threat data from multiple sources saves analysts time by providing up-to-date information on known threats and facilitating the communication of this intelligence within the organization and with external stakeholders.

In addition, **Security Orchestration, Automation, and Response (SOAR)** tools are gaining traction as essential components in security strategies, given the escalating security demands. SOAR technology enables the collection and utilization of data from security operations, defining and prioritizing incident response activities that blend human and machine capabilities for incident analysis and triage [84]. SOAR not only centralizes threat visibility but also automates routine tasks and, therefore, supports and scales the capabilities of human analysts [12]. Comparing SOAR with **Security Information and Event Management (SIEM)**, each has unique capabilities that, when combined, provide a comprehensive approach to incident response [93]. While SIEM systems are adept at managing and analyzing vast quantities of security data to identify threats, they are not as effective in handling the subsequent range of operations required for incident response. SOAR steps in with its advanced automation capabilities and the potential to reduce the need for human intervention in certain incident processes. By integrating SIEM and SOAR, organizations can optimize their security efforts. SIEMs handle the heavy lifting of data ingestion and alert generation, while SOAR systems facilitate the incident response process by automating and orchestrating routine tasks [12].

Academic research in the incident response domain is limited but growing, with studies exploring various challenges and proposing solutions. Shaked et al. [92] address the crucial role of incident response in organizational cybersecurity. They critique the lack of a disciplined approach in current playbook design and representation, which may affect their effectiveness. The authors propose a formal, model-based design approach for cybersecurity incident response playbooks, and introduce a tool prototype. Furthermore, Gurabi et al. derived requirements for the transition to structured security playbooks and their integration with other security tools for incident response, reporting, and automation. They extended the effort by developing a framework for a tool-assisted incident response based on playbooks [4]. Additionally, Schlette et al. [85] conduct an in-depth investigation into incident response playbooks. The research reveals ambiguities in how playbooks are defined and used by practitioners, finding that playbooks often cannot be applied directly across different organizations due to individualized definitions and varying focuses on incident response areas. In the context of response automation, Nespoli et al. [68] discuss the concept of countermeasures and the need for standardized security automation, highlighting issues like scalability and knowledge management. The authors of [75] glean insights from incident response analysts, emphasizing the importance of stakeholder engagement and maintenance concerns in automation, while warning against over-automation that could complicate tasks. Andrade et al. [10] review the literature on security incident processes, while [94] analyzes standards and decision-making gaps in incident analysis. Regarding incident-related models, various approaches are proposed, such as an automatic decision-making model [106], a deep learning-based event classification for SOCs [47], and a data mining approach to predict cyberattacks [50]. Additionally, network-focused research includes the development of systems like the Network Entity Reputation Database System [13] for characterizing network entities and predicting malicious activities. Moreover, Husák and Čermák [51] explore the use of recommendation systems in the context of incident handling and response. Their investigation revolves around the implementation, challenges, and various applications in this field. They particularly emphasize the role of recommender systems in identifying and prioritizing incidents, suggesting appropriate mitigation strategies, and offering real-time insights into evolving incident trends. Further, in [54], authors review automatic incident response solutions, categorizing them using the MITRE D3FEND framework and comparing academic approaches with commercial solutions.

The research contributions reflect a thorough effort to improve incident response through a combination of standardization, automation, and predictive analytics. In the following, we delve into three different elements of incident response in our work. First, we start with the cybersecurity playbook management acknowledging its importance in the current cybersecurity landscape. Subsequently, we examine the role of incident similarity in providing recommendations and enhancing the support of human operators. Lastly, we explore the automation of incident response.

6.1 Cybersecurity Playbook Management

The formulation, management, and sharing of cybersecurity playbooks are necessary for effective incident response and governance in an increasingly complex threat environment. Within the SAPPAN project, a novel vocabulary and process have been formulated to model response and recovery steps for incident response workflow documentation, leveraging semantic Web technologies. This methodology emphasizes the capturing, management, and sharing of knowledge to provide recommendations for manual operators or the automation of actions that reduce human intervention. The methodology and schematics of playbooks provide most of the functionalities that are also provided by the OASIS specification **Collaborative Automated Course of Action Operations (CACAO)** for Cybersecurity [79]. However, the SAPPAN vocabulary follows a more flexible approach in methodology and structure. This flexibility includes modeling confidentiality levels for single resources, which facilitates refined access control, data sanitization processes, and the formulation and dissemination of shareable playbooks. In the development of our vocabulary, we employ established semantic Web standards such as **Resource Description Framework (RDF)**, RDF Schema, and Web Ontology Language 2. It potentially provides the opportunity for easy integration of a knowledge base that follows a similar specification.

Further, SAPPAN introduces a semantic Web-based knowledge-capturing approach for playbook creation, storage, management, conversion, sanitization, and sharing [3]. It includes a tool based on **Semantic MediaWiki (SMW)**, which integrates semantic Web technologies into a MediaWiki knowledge base. As discussed in [69], it is necessary to sanitize the playbooks before sharing. Therefore, our process consists of a playbook sanitizer module, automating the extraction of public or shareable playbooks from confidential response and recovery processes. This proof-of-concept component eliminates manually labeled sensitive data based on the **Traffic Light Protocol (TLP)**. It subsequently generates a version of the playbook that can be disseminated to targeted organizations, specific departments, or security operation levels. Additionally, the system incorporates a playbook converter module. This feature is configured to import playbooks in SAPPAN or CACAO format directly into the management system and to export playbooks to those formats. Also, for sharing the playbooks, a playbook subscriber module enables searching and importing playbooks from the MISP platform and facilitates their distribution. Lastly, the steps within each playbook are schematically modeled and archived within the playbook management system. These steps are visually represented through **Business Process Model and Notation (BPMN)** diagrams, which can be interactively navigated and amended. In the course of the work, a number of response and recovery playbooks for malicious domain detection have been proposed and developed. This also includes generalization and sharing of the playbooks for adoption in other organizations.

6.2 Incident Similarity for Recommendation and Support of Human Operators

When handling a security incident, the goal of a SOC operator is generally to label the incident as indicative of truly malicious behavior (a true positive) or not (a false positive). If the incident represents a genuine attack, response actions are then initiated. In a naive attempt to use ML methods to assist in incident handling, one may try to train a classifier on historical data of labeled incidents and use its predictions to handle future incidents automatically.

However, labeling security incidents is a labor-intensive procedure, and the largest SOCs can handle no more than a few dozen incidents every day. On the other hand, the underlying features of a security incident are complex and heterogeneous: command lines, names of executables, file paths, file binaries, and so on. Therefore, in any realistic scenario, the size of the feature space dwarfs the number of labeled data points, making supervised ML models susceptible to overfitting. This situation is then complicated by the fact that labeling is an inexact procedure performed by security analysts of varying levels of expertise, and thus obtaining ground truth labels is a more inexact science in comparison to other domains such as image recognition.

It is then natural to use unsupervised ML models to assist with incident response. Perhaps the best-known unsupervised ML framework used in cybersecurity is anomaly detection (see, for example, [6, 22, 39]). However, anomaly detection methods rely heavily on having a large sample of clean data from which one can extract a probability distribution corresponding to non-anomalous behavior. Due to our extremely high-dimensional feature space, heterogeneity of the data, and the lingering uncertainty that no sufficiently large sample of incident data can be truly devoid of malicious behavior, modeling the underlying probability distribution of non-malicious behavior appears intractable.

Our ML framework of choice is instead a *similarity model*. Instead of making absolute judgments about individual incidents, we opt to train a model, which makes relative judgments in recognizing when two given incidents are similar to one another. To a security expert handling an incident, this can provide critical context: Given an active incident, one can query the model to discover how previous highly similar incidents were handled and act accordingly. This can increase the overall efficiency of SOC operations.

As part of the SAPPAN framework, we have constructed an incident similarity model whose training and inference algorithms function as follows. First, an incident vectorizer is trained on the past X number of days of incident data, obtained across all client organizations. The vectorized incidents used for training are then stored as part of the model. During inference, an end user, such as a security analyst, inputs a query incident from a given organization into the model, which is then vectorized by the trained vectorizer. Its cosine similarity with all incidents from the same organization in the training data is computed, and the most similar N incidents are returned to the user. Model validation is performed by a team of data scientists and security experts, meticulously going through a number of sample query incidents to verify that the corresponding similarity scores match expectations. To satisfy end-user demands, training and inference are both done rapidly with low-complexity algorithms, which take advantage heavily of the sparsity inherent in incident data.

To preserve data privacy with respect to individual organizations, end users, who may also be security experts employed at the given organization, are only shown similar incidents from their own organization during model inference. On the other hand, the vectorizer is trained on all data from all client organizations, allowing the vectorizer to properly understand individual features in a global context. Another approach would be to train individual models for each client organization; however, this leaves smaller organizations without sufficient data for training their own model without a way to compute meaningful similarity scores for incidents. In this way,

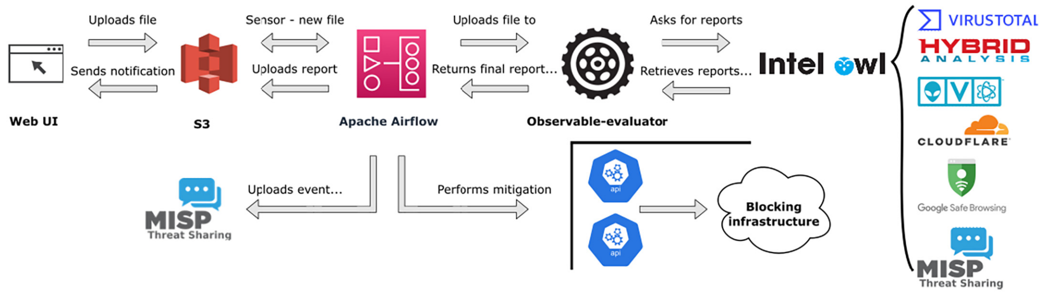


Fig. 4. Malware pipeline consisting of several microservices and tools orchestrated by Apache Airflow [97].

the incident similarity model demonstrates that sharing data across organizations in a way that model outputs still preserve client privacy can benefit SOC operations greatly.

6.3 Incident Response Automation

Responding quickly to an incident is one of the main objectives of security teams to prevent more severe impacts. However, a large amount of analysis is still done manually, and analysts often find themselves repeating the same procedures over and over again, just on different data, which prolongs incident response. An example of such a repetitive procedure is the analysis of suspicious software, where it is necessary to extract its identifiers (e.g., hashes, strings, and IP addresses) and verify in the threat intelligence database whether it is malware. Subsequently, a report should be generated, and appropriate mitigation steps should be taken if it is identified as malware. To demonstrate and validate how all these steps can be automated, we designed and implemented a malware analysis workflow [97], shown schematically in Figure 4.

The system consists of several components, as depicted in the architecture overview given in Figure 4. Here, we provide a short description of the key components:

- *WebUI*: WebUI is a simple React form application. It allows users to upload malware samples for evaluation and to read finished reports.
- *S3*: We use S3-compatible object storage MinIO to store malware samples and reports. The main advantages of MinIO are that it is lightweight, simple to interact with, and has official Docker and Apache Airflow support.
- *Apache Airflow*: Apache Airflow is an open source Python-based workflow and orchestration platform. It contains the orchestration logic of our process, periodically checks new uploads in the MinIO bucket, and sends malware samples to the Observable Evaluator. Furthermore, it obtains analysis results and triggers necessary reporting and mitigation procedures.
- *Observable Evaluator*: Our custom tool that is used for malware analysis. It sends samples to IntelOwl and collects analysis results. The main idea behind this tool is that IntelOwl simply uploads samples to third-party tools and returns their raw results. It performs neither “recursive analysis” (meaning analysis of observables, e.g., IP addresses, in malware sample) nor analysis evaluation (verdict or score).
- *Mocked Blocking Infrastructure*: To demonstrate the mitigation capabilities of the process, we are using a custom mocked blocking infrastructure closely based on the real-world infrastructure. It consists of an IP-blocking API and a domain-blocking API.

The aim of the proposed workflow and implemented prototype is not to undergo a deep analysis of malware behavior but rather to quickly evaluate a suspicious sample and automate the initial response. Moreover, we wanted to test the automation of the whole process, including the mitigation steps and discovering potentially infected machines in the organization’s network.

7 Task: Sharing

Transitioning from local attack detection and incident response to a collaborative perspective requires integrating some form of information sharing. In the following, we introduce some technologies relevant to sharing CTI and introduce sharing scenarios related to collaborative detection and response. We start by briefly mentioning some relevant formats, platforms, and sensitivity signaling approaches to provide a high-level overview of the sharing domain. The breadth of what can become cybersecurity-relevant data and the level of details render it very difficult to introduce a universal overarching standard that would capture all of them. Although there are frameworks that aspire to do so, such as STIX [78], we witness many other formats being used in production, e.g., due to lower complexity. CybOX [76] is used for describing cyber observables and cyber threats across various categories and serves as one of the building blocks for STIX. STIX structures information into key constructs, ensuring comprehensive and expressive threat sharing building on objects and relations. Common Event Expression [67] unifies and structures log and audit data, facilitating universal and machine-readable event records. OpenIOC [63] is a format for describing attacker activities and indicators, supporting extensibility. OpenC2 [77] standardizes the orchestration and automation of cyber defense and can be instantiated by the CACAO [79] format for playbooks. MISP [66] is a widely used platform for sharing intelligence on malware and cyber threats, expressed in JSON, encompassing objects, attributes, and taxonomies. As such, it provides its own data model and format which can be easily extended by adding new objects and taxonomies. MISP also implements distribution levels in its sharing platform; hence it is possible to define who receives the shared piece of information. Also, confidential information can be tagged by approaches such as TLP [43] or the **Information Exchange Policy (IEP)** framework [42].

When aiming to make sensitive data available to other parties, data sanitization can be considered as a way to prevent the leakage of sensitive information to the receiving parties. While anonymization (as a way to sanitize personal information) has been extensively studied in the past (e.g., [58, 62, 95, 110] for relational data and [52, 61, 109] for graph data), respective solutions for technical information have been more scarce, but format-specific approaches have, for example, been proposed for IP addresses [104] and packet traces [80]. Other approaches such as differential privacy [35, 36], **Secure Multi-Party Computation (SMPC)** [108], and **Homomorphic Encryption (HE)** [2] apply privacy-enhancing modifications to data processing tasks rather than the data itself. We want to emphasize that cybersecurity use cases may deal with sensitive information that is not necessarily of a personal nature. Using the term *privacy* in this context may hence conflict with privacy definitions that are strictly related to personal information. However, a clear distinction between these categories is complicated by the circumstance that it is not always clear whether a specific piece of technical information could ultimately be attributed to a person. Nevertheless, many approaches in the category of privacy-enhancing technologies can conceptually also be used to protect sensitive non-personal information. This is because in both cases, the ultimate goal is to avoid sharing information from which undesired consequences may arise. We thus focus on the protection of information for which sharing beyond a certain trust boundary (e.g., on an organizational level) could lead to undesired consequences.

In the following, we consider three different aspects of sharing in a cybersecurity setting. First, we consider sharing and federation for cyber threat detection, focusing on sharing federated learning updates to collaboratively train DGA detection classifiers. We then consider privacy and security risks associated with the training data used in the DGA detection use case, and briefly list the privacy-enhancing approaches that we have evaluated as part of our validation. Lastly, we outline our work done to enable sharing of cybersecurity playbooks.

7.1 Sharing and Federation for Cyber Threat Detection

The traditional approach of sharing for cyber threat detection is to share IoCs, such as a malware binary or an IP address of a C2 server. Our aim is to broaden the scope of sharing data between multiple organizations supporting privacy. We aim not to share the raw data but rather metadata such as the classification models. In such a case, the classification models become new IoCs, for example, represented by a neural network or a decision tree.

Once we are able to share models, we can establish various ML training schemes across organizations, such as federated learning. In federated learning, every party that is participating in the training of a global model first acquires the current model, which can also be a randomly initialized model at the beginning. Then, each party individually improves the current model using its private data and subsequently summarizes the changes to the model as a small, focused update. Every party distributes its model update and averages it together with the updates of all other participants. Using this federation step, every party can now apply the collaboratively computed update to the shared model to improve it. This is an iterative process, and multiple federation steps are performed until the global model reaches a maximum regarding classification performance on a certain validation set. All participating parties train a neural network classifier collaboratively by applying averaged model updates to a shared global model.

We investigate two different approaches for federated learning that differ in their type and amount of federation steps: federation after model convergence and federation after each training epoch. In the first case, either a randomly initialized or a pre-trained neural network classifier is distributed among all participating parties. This model is used as a starting point for all collaborating parties. The pre-trained global model can be trained based on a publicly available dataset (e.g., Alexa or Tranco top domain names for benign training samples in the DGA classification use case) so as not to leak sensitive information. All parties then continue the training of the initial classifier using their own private data. After each locally trained model converges (i.e., it reaches a maximum regarding classification performance on a validation set), all parties compute the updates in relation to the initial model. The updates equal the difference of the neural network classifiers' weights to the initial model. Subsequently, all updates are merged by averaging and applied to the initial model, which yields the final global model.

The second case, federation after each training epoch, is similar to the previous one but differs in the type and the number of federation steps. Instead of training the initial model locally up to its convergence, each party shares the computed update after each training epoch. Then, similarly, all updates are merged by averaging and applied to the initial model, which yields the global model for the next training epoch. All parties then continue the training of the updated global model for another epoch. This process is continued until the global model converges. The chosen approach for federated learning influences the global model's classification performance and the level of privacy provided.

The federated learning scenario is facilitated by the MISP platform with all the implications. As such, the sharing parties can define distribution levels such as sharing within the local MISP instance or sharing synchronized with other MISP instances (if there are partners willing to share their models and capable of processing the data model proposed for sharing ML models). The shared data must include the following attributes as a minimum:

- Classification Model: The actual classification model contains zero-weights for architecture description but the initial model must be randomly initialized in a global way.
- Model Formats: ONNX, HDF5, XML, numpy array (for model updates).
- Use Case: DGA classifier global init, DGA classifier update, DGA classifier final.
- Unique Training Identifier: An identifier used to synchronize the training procedure, e.g., a hash of the previous round.

The latest shared model is, in fact, an IoC and the organizations involved in the sharing can apply it in their cybersecurity analytic frameworks to discover suspicious events, such as the detection of AGDs in DNS **Non-eXistent (NX)** domain replies.

7.2 Cybersecurity Data Sanitization for Privacy- and Security-Aware Sharing

Whenever data are shared, the question of potentially negative consequences in doing so arises. Especially in the context of cybersecurity, concerns of unintentionally disclosing information that may benefit malicious activities have to be addressed. In the context of DGA detection, it is to be noted that only the benign samples of the training dataset (the collected NX domains) are considered to be sensitive, as malicious samples are available as OSINT. The benign NX domains, however, can be considered to be both security and privacy critical. From the security perspective, NX domains may leak information about software used in the network, potentially revealing the use of outdated versions with known security issues. When containing user-queried domains, the set of NX domains can further provide information about browsing behavior in the network. Such information may further provide information about the organization, in which the NX domains have been collected. This combination of known vulnerability and identifiability of the respective target poses a security risk. Another security concern is introduced by misconfigured devices, which try to connect to services that are no longer online, resulting in a NX domain request. In such a case, an adversary may take advantage of this situation and register the respective domain to mimic the respective service.

As such, the benign training data samples in the DGA detection use case should be protected when considering a scenario that involves sharing. Beyond addressing the above mentioned security concerns, the samples may also have to be protected to avoid the leakage of personal information, or to comply with organization guidelines and legal obligations such as data protection regulations or customer contracts.

Even though federated learning allows to benefit from diverse training data sources, it requires each contributing party to participate in the training process. Organizations with a lack of potent hardware may, hence, become a bottleneck in the training process. But also aside from requirements for participating in the training process, additional measures may need to be taken to address privacy concerns surrounding training data extraction from model updates. While respective concerns can be addressed by secure aggregation of model updates [17, 40] or the application of local differential privacy during training [98, 103], these approaches commonly take a toll on performance or model utility [59]. We thus also evaluated settings beyond federated learning. Specifically, we considered two approaches for training data sanitization as well as different cryptographic schemes for privacy-preserving **Classification-as-a-Service (CaaS)**. The sanitization of training data allows to create and utilize larger and more diverse training datasets from multiple sources without the requirement that every data provider actively needs to participate in the training process. This also allows for the asynchronous collection of training data, and provides the receiving organization more flexibility in how the data can be used, for example, in regard to the specific type of ML model that is trained on it. While sanitized training data can also be used to hide sensitive information in both a **Machine-Learning-as-a-Service (MLaaS)** and a CaaS setting, the evaluated cryptographic schemes provide strong formal guarantees, which are commonly more difficult to obtain for sanitization approaches. Specifically, we have considered the following approaches:

(1) We transferred a similarity-preserving encoding technique from the area of privacy-preserving record linkage to the DGA detection scenario for the purpose of training data sanitization [70, 71].
(2) We performed a privacy analysis of the FANCI feature extraction approach [90] to evaluate, whether the domains can be reconstructed based on their respective extracted feature vectors, and hence whether the FANCI feature extractor may be used as a sanitization approach in the DGA detection scenario [49].
(3) We evaluated existing frameworks that implement different SMPC and HE schemes as cryptographic approaches in the CaaS setting to protect samples during inference [32].

7.3 Cybersecurity Playbook Sharing

The academic and practical implications of sharing cybersecurity playbooks hold significant promise for the evolution of CTI ecosystems. Collaborative engagement in capturing and disseminating cybersecurity playbooks and their steps is important for the automation of response and recovery procedures [72]. In an effort to improve CTI exchange and facilitate collaboration, we essayed a requirement analysis [107] of the sharing platforms MISP, STIX, and OpenCTI. Further, we investigated the MISP architecture in more detail as the selected sharing platform. This investigation contains a systematic survey of MISP objects, tags, galaxies, attributes, and types, with the objective of developing a coherent data model for cybersecurity playbook sharing. The initiative also aims to create a sharing interface integrated within MISP's existing framework. In a methodological approach, we mapped the playbook along with its associated metadata to the existing MISP data model. This mapping exercise served as a critical point in identifying synergies between different playbook data structures and MISP's modular architecture. We implemented the MISP CACAO object, incorporating it into the official MISP repository. This effort not only extends the functionality of the MISP platform but also validates the feasibility of the newly developed data model and sharing interface.

8 Validation

Our validation addresses multiple facets of the SAPPAN architecture and the presented tasks. We evaluate improvements from a technical and quantifiable perspective where possible. We complement the technical evaluation with an opinion-based evaluation as expressed by a team of domain experts who assessed the SAPPAN framework on the specific use cases. In the following, we summarize our results and key findings per task (local attack detection, incident response, and sharing), and provide pointers to more detailed descriptions where available.

8.1 Local Attack Detection

8.1.1 Malicious Domains Detection. Our work on the detection of malicious domains focused on two use cases. First, we summarize our findings on phishing detection using CT logs, followed by our results on DGA detection.

Phishing Detection via CT Logs. In validating our proposed methods for local attack detection, our initial focus was on detecting malicious domains, specifically through phishing classification in CT logs. Detailed results of this evaluation can be found in [31]. The following text summarizes the main results. Our data processing pipeline is designed to efficiently collect, standardize, integrate, refine, and annotate data from various sources. This flexibility allows our methodology to handle complex datasets of different structures and origins effectively. Using this pipeline, we have compared the developed classifiers with those described in existing literature. Regarding computational efficiency, we benchmarked our system's performance by processing a week's consolidated CT logs, completing the task in approximately one day. To illustrate, a single pipeline operation (which integrates four distinct meta classifiers) was executed across 24 cores of an Intel Xeon Platinum 8160 processor running at 2.1 GHz, which required about 55 GB of RAM. Consequently, we anticipate that our framework is well-prepared to quickly assimilate and process the significant number of certificates added to CT logs in real time. Additionally, our design emphasizes scalability to ensure optimal performance as computational capacities are increased.

DGA Detection. In addition to the analysis of CT logs, we also focused on evaluating DGA classifiers in the real world as part of the methods for local attack detection. We trained 20 classifiers to classify an unfiltered 1-month recording. A detailed description of the results can be found in our paper [34], and the following text summarizes the main results. In our month-long observation, we thoroughly sifted the data against DGArchive [81], revealing 13,870 unique domains created by six separate DGAs. These domains are readily identifiable through rudimentary blocklisting methods. Remarkably, 17 of the 20 classifiers we used could accurately identify all known harmful domains, resulting in an average **True-Positive Rate (TPR)** of 99.997%. Subsequent analysis of the residual samples, flagged as positive by our classifiers, was undertaken to unearth novel DGAs. By implementing a semi-automated strategy, we scrutinized these marked samples to detect previously unknown AGDs. We used a comprehensive clustering approach that included domain length, constituent characters, top-level domains, query frequency, temporal occurrence span, Shannon entropy, and the presence of English words to classify these domains. Leveraging the combined power of DGArchive, domain expertise, and thorough manual examination, we sorted 5,833 enigmatic domains into eight separate clusters. Interestingly, six of these clusters likely correspond to unidentified DGAs, one matches with an elusive seed of the Bamital DGA, and another aligns with the signature of the Conficker DGA. Notably, the Conficker samples generated earlier than their expected operational timeframe explain their absence from the historical DGArchive records.

As part of our focus on malicious domain detection, we further explored the possibilities of interpretable visualization of deep neural networks for DGA detection. We used seven cybersecurity experts (CSIRT team members, cybersecurity data analysts, and visualization experts) for initial testing. The technical report [14] provides a detailed methodology and further evaluation description. The initial findings indicate that the suggested visualizations could lead to a more consistent trust in users' decisions and find value in the visualizations. In behavior-related tasks, participants develop a more intricate understanding of the model and frequently cite the analysis visualizations as having a substantial influence on their decisions. However, those utilizing the visualization tool required a significantly longer time to finish any task and needed an extended training period compared to the control group, likely due to the more intricate nature of the interface overall. Regarding expertise, the current data imply that beginners exhibit a higher average confidence in their decisions and explanations. Beginners in visualization also report a higher degree of usefulness than their expert counterparts, while the difference in reported usefulness for ML beginners and experts is less noticeable.

8.1.2 Anomalous Behavior Detection. In addition to detecting malicious domains, we focused on anomalous behavior detection, primarily on application profiling and process mining. In the case of fingerprinting based on DNS traffic, we validated the capabilities of the proposed approach on both simulated data and real data [91]. In our empirical analysis, Windows 10 was detected on roughly 11% of all hosts that initiated at least one DNS query. Among hosts where no particular application was identified, almost half (48%) sent less than 100 DNS packets in total. It is crucial to note that our datasets included a variety of devices, such as desktops, laptops, and mobile phones, but our analysis rules were specifically designed for Windows 10. Interestingly, our investigation revealed only a handful of fingerprints linked to crypto-mining activities. This scarcity could be due to the decreased presence of such applications during the observation period or potential variances in application behavior between our experimental environment and the broader dataset. Notably, our rule creation primarily resulted in a single label for Nicehash and three labels for Easyminer. Applications with more detections typically displayed a more comprehensive array of unique domain queries, thus providing a more diverse set of labels for analysis. Nevertheless, our tool exhibited notable effectiveness in identifying numerous unique fingerprints within the dataset.

To evaluate our tool for anomaly detection on application behavior using process mining, we used a simulated dataset consisting of *RED-benign* captured over 51 days on a simulated environment without any attack data and *RED-attack* captured during 1 day red-teaming experiment [48]. Our validation began by using the test dataset as input for our tool, in conjunction with the generated models, to ascertain the accurate depiction of benign application behaviors on the hosts. Following this, we introduced the attack dataset into the system to assess whether the trace checker would highlight increased errors indicative of anomalous activities. The results are presented in Table 3. Upon examining the test dataset, our tool demonstrated a precision of 97.22%. The discrepancies mainly originated from instances where no starting point was identified. This can be attributed to specific processes being operational before the initiation of our data capture. On the other hand, with the attack dataset, a higher rate of discrepancies was noted, with only 71.15% of events conforming to the established model. Of the remaining 28.85%, 6.59% were due to missing start points, which can be classified as false positives, as previously explained. Most of the discrepancies resulted from unexpected successors, suggesting that the subsequent events occurred earlier than expected within the trace or introduced entirely new sequences.

8.2 Incident Response

8.2.1 Playbook Management. The playbook management tool that allows a user to create, modify, and manage playbook steps and their relations was rated by the surveyed experts as very useful to help not only to create but also to understand the playbook steps via the BPMN graph representation. The SAPPAN playbook management

Table 3. Errors for the Test and Attack Datasets Compared to the Benign Model, Adapted from [48]

Tool output	**RED-test vs. RED-train**	**RED-attack vs. RED-benign**
true	97.22%	71.15%
true, but insertions	0%	4.12%
multiple fingerprints not part of the fingerprint matrix	0%	6.32%
successor is not correlated to the fingerprint	0%	10.16%
no startpoint found	2.78%	6.59%
endpoint not valid	0%	0.55%
startpoint not in the fingerprint matrix	0%	0.27%
endpoint not in the fingerprint-matrix	0%	0.82%

toolkit, incorporating SMW, provides functionality crucial for sanitizing, converting, presenting, and sharing playbooks. This involves a simple playbook sanitizer module that automates the extraction of public or shareable playbooks from confidential response and recovery processes, effectively removing labeled sensitive information. Complementary to this, the playbook converter and subscriber modules facilitate the import and export of playbooks in various formats and their distribution within the MISP platform. The playbook presentation module represents the response and recovery steps into BPMN graphs for interactive management. In our opinion-based evaluation, all surveyed experts rated the improved understanding of playbook steps and recommendations between 7 and 9 on a scale from 0 (does not help) to 10 (nothing better can be done), with an average score of 8.2.

8.2.2 Incident Similarity. The domain experts rated the ability to gather similar incidents and how they were handled as very useful. On average, they rated it between 7 and 8 on a scale of 0–10 of usefulness. They noted that the incident similarity improves the context knowledge of an incident handler and that this context knowledge, based on similar incidents handled in the past, can be used to reveal false-positive cases and support decisions on what should be the next steps when handling the incident based on the steps taken to deal with similar incidents.

In addition to the subjective validation of the incident similarity model as described above, we also validated the model quantitatively by computing the empirical distribution of similarity scores and confirming that it matched the security experts' expectations. To conduct this experiment, we sampled 5,000 similarity scores at random from 30 days worth of incident data, and plotted the resulting distributions in Figure 5.

The left plot in Figure 5 shows the distribution of all 5,000 randomly sampled similarity scores. Clearly, the distribution is heavily skewed towards 0. This is as expected, as two randomly chosen incidents will generally exhibit no meaningful similarity. The presence of a small, but noticeably non-zero, proportion of similarity scores, which are in the 0.1–0.2 range is explained by the presence of, for example, executables such as `cmd.exe` whose presence is ubiquitous in the incident data but signify little in terms of actual similarity.

In the right plot in Figure 5, we restrict our distribution to those similarity scores, which were at least 0.7. As customers, partners, and security experts are mostly interested in seeing pairs of incidents, which the model deems *are* similar, this represents a sample of model output, which would actually be shown to end users. As one can see from the plot, the distribution is concentrated close to 1.0, meaning the model is deeming a large proportion of incidents to be nearly identical in nature. Identifying such swaths of extremely high similarity scores enables mass processing of and response to incidents. Lastly, we note the spike in the distribution around 0.88. This was caused by two large collections of incidents that were validated by experts to be highly similar to each other.

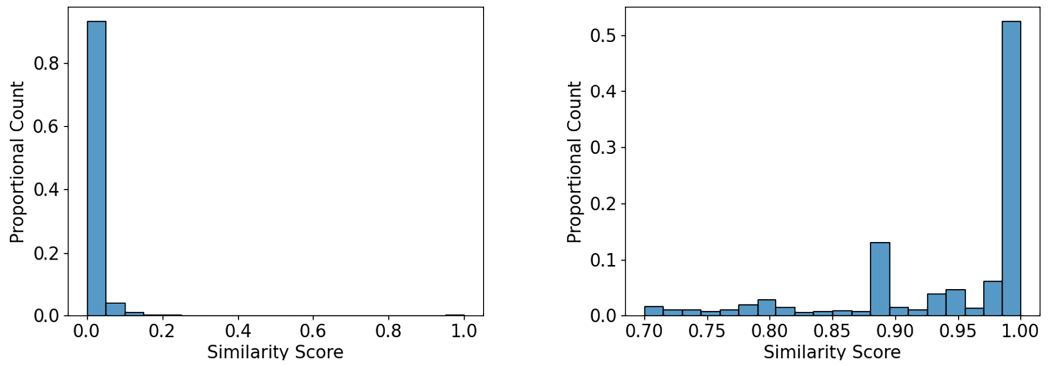


Fig. 5. Empirical distributions of incident similarity scores. On the left, the distribution of all similarity scores, and on the right, the distribution of those similarity scores which were at least 0.7.

8.2.3 Incident Response Automation. We have measured the malware analysis workflow to validate the benefits of incident response automation. We asked four professionals from a university CSIRT, responsible for daily operations and incident response, to go through the malware response process. Each analyst was given the same samples to process. Sample 1 was an MS Word document created by the test's authors to serve as a benign sample. Samples 2 and 3 were taken from the theZoo repository [105] and are actual malware samples. The goal was to decide whether a sample is malicious, as sometimes users report suspicious files that are benign. The analyst was asked to obtain network IoCs from the malicious samples and for each IoC to decide whether it is malicious or not. The next task was to block communication with malicious IoCs at the organization level. Finally, the analysts were asked to identify potentially compromised machines in the network by searching for the communication in network traffic records.

The results of the comparison of manual and automated malware analysis are in Table 4. The time measurement started with downloading the sample from the ticketing system, and the times in the table correspond to the times when the analyst finished the given task. In addition to time, we discussed the process with analysts and collected their feedback. They acknowledged that using the malware evaluator can reduce possible mistakes as a human can overlook or forget an IoC. They also mentioned as an advantage that they could upload the suspicious file and wait for the analysis results while doing other work instead of the manual analysis requiring their full attention the whole time. The validation also showed that the main challenge of automated processing is the availability of high-quality open source intelligence data, as several regular services were marked as malicious during the testing.

8.3 Sharing

8.3.1 Federated Learning. The sharing of ML models enables a collaborative approach for training without exchanging the data itself. In principle, this allows to utilize data that cannot be centralized (e.g., due to legal reasons). We demonstrated feasibility of federated learning in the DGA detection use case by training classifiers locally on data of a single institution and comparing this baseline to classifiers trained in federated fashion on data of four institutions [33].

For creating training and test datasets, we obtained malicious AGDs from one data source, and benign NX domains from four data sources. The malicious domains have been sampled from DGArchive [81], an open source intelligence feed that contained approximately 126 million unique domains generated by 95 different known DGA families at the time of sampling. The benign NX domains have been collected in four organizations (two universities, one association of universities, and one company). Using these data sources, we constructed four training and four test datasets, one for each source of benign samples, respectively. Each training dataset

Table 4. Comparison of Manual and Automated Processing of Potentially Malicious Sample Files

			VirusTotal	**Hybrid analysis**	**IoCs collection**	**IoCs evaluation**	**IoCs blocking**	**IoCs traffic**	**Total time**
A1	**S1**	**M**	00:28	01:17	08:55	x	x	x	**08:55**
		A	00:25		01:29	x	x	x	**01:29**
	S2	**M**	00:27	01:18	03:52	06:20	07:04	08:32	**08:32**
		A	00:31		02:02	02:02	02:02	02:02	**02:02**
	S3	**M**	00:24	00:59	02:40	03:45	05:01	06:10	**06:10**
		A	00:15		02:17	02:17	02:17	02:17	**02:17**
A2	**S1**	**M**	00:20	01:07	01:42	x	x	x	**01:42**
		A	00:24		02:28	x	x	x	**02:28**
	S2	**M**	00:21	00:46	02:12	03:04	x	x	**03:04**
		A	00:14		01:58	01:58	01:58	01:58	**01:58**
	S3	**M**	00:15	01:02	01:53	02:20	04:01	05:24	**05:24**
		A	00:13		01:45	01:45	01:45	01:45	**01:45**
A3	**S1**	**M**	00:20	00:56	01:20	x	x	x	**01:20**
		A	00:20		01:24	x	x	x	**01:24**
	S2	**M**	00:12	00:30	03:15	04:29	x	x	**04:29**
		A	00:16		02:43	02:43	02:43	02:43	**02:43**
	S3	**M**	00:10	00:26	01:37	01:57	02:53	03:30	**03:30**
		A	00:12		02:52	02:52	02:52	02:52	**02:52**
A4	**S1**	**M**	00:54	02:21	02:41	x	x	x	**02:41**
		A	00:48		01:51	x	x	x	**01:51**
	S2	**M**	00:16	01:24	02:24	04:31	x	x	**04:31**
		A	00:45		01:56	01:56	01:56	01:56	**01:56**
	S3	**M**	00:15	01:27	02:08	03:40	04:38	05:38	**05:38**
		A	00:35		01:52	01:52	01:52	01:52	**01:52**

The bold column headers represent the different steps to process a potentially malicious file. A#, Analyst; A, automated analysis; M, manual analysis; S#, sample.

contained 446k samples, and each test dataset 222k samples. Each of these datasets had a 50/50 label distribution (malicious/benign) in regard to the DGA classification task.

To establish the baseline for local training, we performed five repetitions of training and evaluating a classifier for each combination of training and test dataset. In the following, we focus on the results of our ResNet-based classifiers developed in [34]. We took the average TPR and average FPR of these 80 evaluation runs to establish the local baseline.

For federated learning, we evaluated different setups. In the following, we present the result of training the ResNet-based classifier with a randomly initialized global model and federation after each epoch. We performed 5 repetitions for each test dataset and each of the 11 possible combinations of participating organizations. Again, we took the average TPR and FPR of these 220 evaluation runs.

The results are given in Table 5. A more detailed description of this evaluation as well as additional experiments can be found in [33].

Table 5. Improvements Gained by Federated Learning for the ResNet-Based Model Developed in [34], as Evaluated in [33]

Approach	TPR	FPR
Local	0.99844	0.01640
Federated	0.99935	0.00913

The comparison of local and federated schemes demonstrates a minor improvement in the TPR in favor of the federated approach. More importantly, the FPR was reduced by 44% in the case of federated learning. This is a significant improvement when translated into absolute numbers, as the number of non-existing resolutions can easily reach thousands per day in large organizations. The performance overhead caused by federated learning was represented in the majority by the longer time to train the classifier. The CPU and memory utilization was approximately the same for both approaches, but it took three times longer to finish the federated learning process (approximately 9 minutes) compared to local training (3 minutes). Communication and synchronization over the sharing platform caused the major overhead of federated learning.

8.3.2 Data Sanitization. Considering scenarios beyond federated learning, we have investigated two approaches for data sanitization (Bloom encodings and FANCI features) as well as the use of existing frameworks implementing different cryptographic approaches for CaaS. In the following, we briefly present our key findings evaluating these three approaches in the binary DGA detection use case (benign/malicious). The key benefits and shortcomings of the respective approaches are summarized in Table 7.

Training Data Sanitization via Bloom Filter Encodings. As the first approach for sanitizing training data, we considered a similarity-preserving encoding approach using Bloom filters. This encoding procedure has originally been used for the purpose of privacy-preserving record linkage [88], but provides properties that are conceptually suitable for ML use cases [70].

In regard to the general threat model, we considered a collaborative setting in which the encodings get shared with another party. We assume this party to be honest-but-curious with the goal of decoding the received encodings. In this context, two primary cases have to be distinguished: settings in which the encoding parameters remain secret, and settings in which they are accessible to the adversary. We have analyzed the threat associated with the former case more closely under consideration of a setting in which the encodings are sent to an honest-but-curious MLaaS provider for training and classification [71].

We evaluated the Bloom encoding approach using different values for the encoding parameters and different modes of encoding (basic, hardened through randomization [37, 89], and hardened through diffusion [11]) by training featureless CNNs on the respectively encoded versions of the same DGA dataset. The results are given in Table 6. For the basic and randomized encoding modes, a detailed description of the experiment can be found in [70]. The evaluation of the diffused encoding mode has not been published before, but uses the same dataset and experiment setup as in [70]. For the creation of the diffusion sets, the respective algorithm proposed in [11] has been used. The values of the basic encoding parameters have been adjusted in a privacy-driven fashion to suit the ML use case, prioritizing the protection of samples over model utility.

While we observe that the encoding approach has a notable negative impact on the TPR of the trained models, the impact on the FPR in absolute numbers is significantly smaller. As to be expected, applying hardening approaches further impacts the model's utility. Especially the use of diffusion greatly impacted both the TPR and FPR of the model (despite a relatively small diffusion set size of $t = 3$) in the case of 128 bit Bloom filters. The application of randomization had a smaller negative impact, providing an FPR close to the one of the basic Bloom encoding approach. Choosing shorter Bloom filter sizes (here 64 bit instead of 128 bit) increased the negative impact of the randomized approach on both TPR and FPR, but disproportionately affected the FPR.

Table 6. Comparison of the TPR and FPR for CNNs Trained on Differently Encoded Versions of the Same DGA Dataset (101,866 Samples with a 50/50 Label Distribution)

Basic encoding parameters	**Hardening approach**	**TPR**	**FPR**
Cleartext baseline	-	0.99545	0.00030
$l = 128, k = 2, q = 2$	None	0.94726	0.01343
$l = 128, k = 2, q = 2$	Randomization ($f = 0.4$)	0.87944	0.01701
$l = 128, k = 2, q = 2$	Diffusion ($t = 3$)	0.83848	0.05319
$l = 64, k = 2, q = 2$	None	0.90951	0.00754
$l = 64, k = 2, q = 2$	Randomization ($f = 0.4$)	0.84423	0.06670
$l = 64, k = 2, q = 2$	Diffusion ($t = 3$)	0.82438	0.05163

Parameters: l is the Bloom filter length, k the number of hash functions, q the q-gram length, f the randomization parameter of RAPPOR's randomized response step, and t the size of diffusion sets. The TPR and FPR of the cleartext baseline, basic encodings, and randomized encodings are based on the experiments carried out in [70].

Table 7. Key Benefits and Shortcomings of the Different Privacy-Enhancing Approaches Observed in the DGA Detection Use Case Based on [32, 49, 70, 71, 90]

Approach	**Benefits**	**Shortcomings**
Bloom encodings	—Provides many parameters to fine-tune the encodings for a desired privacy-utility tradeoff —Computationally efficient —Easily transferable to other use cases (as long as samples are roughly of similar length) —Can conceptually be used to share sanitized samples —Can be used in the MLaaS setting to hide the actual ML task from the MLaaS provider	—Selection of parameter values requires good understanding of the procedure —Privacy-driven selection of parameter values can have a notable negative impact on classification accuracy —Privacy guarantee not as strong as for cryptographic approaches —Sharing sanitized data for independent use requires receiving parties to know the encoding parameters to map other samples to the same space
FANCI features	—Very high utility —Computationally efficient —The feature extraction procedure does not seem to be efficiently reversible in general —Can conceptually be used to share sanitized samples	—Protection against other threats is unclear (e.g., membership inference, extraction of statistical information) —Changes to feature set require reconsideration of the reversibility property (as potentially required when applied to a different use case)
SMPC, HE	—No negative impact on the quality of classification results (if model is not notably simplified) —Strong guarantees for protecting samples in the CaaS setting —Allows to train on and classify unaltered samples without revealing them to collaborators	—Excessive communication cost (even with model simplifications) —Computationally intensive due to the complexity of cryptographic operations —Does conceptually not allow to share sanitized samples for independent use

For the diffusion approach, however, the shorter Bloom filters surprisingly had no notable impact. We expect that the model utility could be increased by using longer Bloom filters and more hash functions, but at the cost of potentially increased vulnerability to decoding attempts, e.g., via pattern mining attacks [23, 24, 100]. To evaluate the protection provided by the parameter values used above, we have performed a respective analysis considering basic and randomized Bloom encodings for DGA detection in the MLaaS setting [71]. This analysis did not find notable information leakage, and applied decoding attempts through pattern mining attacks remained unsuccessful.

Training Data Sanitization via FANCI Feature Extraction. In addition to obfuscating samples via Bloom encodings, we have also considered the privacy properties of feature vectors extracted via the FANCI feature extractor [49, 90]. As the FANCI feature extractor has originally been introduced to improve classification results in DGA detection, it is not expected to negatively impact utility when applied as a sanitization approach in this use case. The results of a respective utility evaluation can be found in [90]. The use of extracted features instead of cleartext samples, however, also provides some level of obfuscation. To evaluate the degree to which the FANCI feature extraction process is reversible, and hence whether a cleartext sample can be reconstructed from its extracted FANCI features, we have trained recurrent ML models on three different benign DGA datasets to learn this sample reconstruction task. A detailed description of our experiment can be found in [49]. We used the Damerau-Levenshtein distance metric [30] to quantify for each reconstruction how many operations (character substitution, character insertion, character deletion, transposition of adjacent characters) are at least required to construct the actual samples from the reconstruction output by the model. To account for differences in sample length, we additionally considered a normalized version of this distance, which divides the Damerau-Levenshtein distance by the length of the actual sample or the reconstruction, depending on which is longer. Considering the Damerau-Levenshtein distance and its normalized version for each combination of the three datasets (for the purpose of training and reconstruction), the most successful reconstruction attack still required an average of 13.66 operations (with a normalized value of 0.46), and in the least successful case an average of 72.94 operations (with a normalized value of 0.75) to retrieve a cleartext sample from the reconstructed one [49]. This indicates that reliably reversing the FANCI feature extraction process in general is not possible.

Nevertheless, the result does not guarantee that the use of FANCI features cannot leak any potentially sensitive information. For example, the protection against membership inference attacks on datasets of FANCI features and the extraction of statistical information, which may reveal sensitive information about the origin network, are not addressed by this evaluation. It should also be considered that changes to the set of extracted features, which might be required when a different use case is considered, may not provide the same properties in regard to general reversibility.

CaaS via SMPC/HE. Specifically for the CaaS setting in which a model owner trains a model on their own data and classifies samples of other parties as a service, we have evaluated different frameworks that implement cryptographic approaches, which allow to keep the samples hidden from the service provider. The detailed description of this evaluation can be found in [32]. While also the above mentioned sanitization approaches could be used in the CaaS setting by performing both training and inference on sanitized samples, the cryptographic approaches provide strong formal guarantees for protecting samples without sacrificing model utility. We have evaluated the performance of four existing frameworks, of which three implement SMPC approaches (PySyft [83], TF-Encrypted [28], and SecureQ8 [29]), and one implements a hybrid approach that combines SMPC and HE (MP2ML [16]). PySyft and TF-E implement 3-party protocols, while MP2ML implements a 2-party protocol. SecureQ8 provides both 2- and 3-party protocols.

We evaluated these frameworks with four state-of-the-art models (three deep learning models and one feature-based approach utilizing FANCI) on the same DGA dataset in the binary DGA classification task. The inference latency when applying the evaluated frameworks out of the box was prohibitive. Even though we achieved a reduction in inference latency by up to 95% and a reduction in communication overhead by up to 97% through model simplifications at an acceptable reduction in classification accuracy by less than 0.17%, the resulting inference latency per sample still showed to be too high to be applicable to real-world DGA detection scenarios [32]. While still over the desired inference latency of 405 to 430 µs, the evaluated 3-party protocols showed to be significantly more performant than the 2-party protocols, but it remains unclear how to integrate a third party into the CaaS setting in a meaningful way.

8.3.3 Playbook Sharing. Cybersecurity playbooks represent another specific type of data that is shared within the SAPPAN framework. We proposed and implemented a dedicated playbook-sharing object and rendered it publicly available through the official MISP repository. The object is fully compatible with the CACAO standard and is ready for any additional playbook formats and versions. The universality of the object itself is due to its straightforward design. The object consists of metadata describing the playbook properties and the playbook itself in the original format. The receiver is always capable of reading the metadata and can thus figure out the format of the playbook. This allows the receiver to identify, whether the playbook itself is in a desired format. Additionally, the MISP security playbook object can be tagged with additional context information such as MITRE (e.g., an attack pattern) to provide additional contextual information, or IEP/TLP to signalize the level of sensitivity.

In order to receive feedback from the domain experts, we asked how well the playbook sharing via MISP supports the response and recovery process in their organizations. The experts agreed that sharing the playbook in a standard format helps to receive and apply the playbooks. However, they noted that playbook sharing is primarily useful for similar organizations. If the environments of organizations are different, playbooks need to be adapted manually to the local needs to improve the response. The experts also mentioned that the CACAO playbook standard is relatively new and has not yet been adopted, as it was introduced in 2021. Additionally, the SAPPAN effort resulted in a joint publication with the CACAO committee [64]. This documents the initiative work carried out in the field of CTI sharing by the SAPPAN project, which not only introduced playbook sharing via MISP but has also made further recommendations for the enrichment of threat intelligence collaboration, particularly in the context of playbook sharing.

9 Discussion

In the following, we discuss the implications that each of our results have from three different perspectives per result. First, we provide a general discussion of the respective validation result. Secondly, we discuss the implications the result has in regard to the challenges and potentials of collaboration and automation as identified in Tables 1 and 2. And lastly, we align the result with components in the SAPPAN reference architecture as described in Section 4.

Detection of Malicious Domains and Anomalous Behavior. Local threat detection is the initial step in ensuring the security of the protected infrastructure and provides input for both incident response and collaboration through information sharing. Currently, there are a large number of different tools and techniques that can be used for this purpose. To demonstrate some of these techniques, we have chosen to detect malicious domains and anomalous behavior using ML techniques, which are now typical for modern threat detection. Evaluation of these techniques has shown their potential not only as stand-alone detection mechanisms but also as a suitable source of shared data when they provide sufficient accuracy.

The selected techniques are intended to show how the identified objectives can be approached and what potentials arise from them. In the case of DGA detection, we have proposed a tool that allows the use of different ML algorithms and compare them with each other, which has a potential for incident response (C.C5), where the analyst can not only use the result of the detection itself but also compare this result with other techniques to gain further information about the detection validity. Together with the proposed anomaly detection techniques, our methods address the potential of preemptive defense (C.P1), where ML allows us to detect novel kinds of attacks effectively. In addition to the detection mechanisms, we also focused on supporting the analysts in developing novel methods. We presented a tool for understandable interpretations of DGA classification, demonstrating how to increase confidence in ML model interpretation even in the face of a complex problem domain (C.P4). Consequently, increasing trust also plays an essential role in the possibilities of information sharing and further use.

Local detection mechanisms form the initial part of the SAPPAN reference architecture. Based on the outputs of these mechanisms, incidents are created and further managed through case management. At the same time, the outputs of local detections form the primary information shared through the sharing system, where local attack detection methods of other organizations may utilize it.

Incident Similarity. Our validation of the incident similarity-based methods has shown that incident similarity clustering is instrumental for the mass processing of similar incidents. According to our empirical experiments as well as according to our domain experts, it is a pivotal tool to automatically identify the FPs that fall within the cluster of incidents previously tagged as false positives within the context of similar organizations.

From this viewpoint, the incident-similarity clustering aligns effectively with the potential for faster processing and resolution of incidents (A.P2). The method utilizes the knowledge of human analysts based on their previous decisions on whether similar incidents were discerned as false positives. It serves as a bridge between human expertise and automated processes to reduce the workload on the human operators (A.P1). The clustering of similar incidents and their visualization also offers the opportunity to improve human–machine teaming (A.P4) by observing and analyzing the clusters of similar incidents across multiple organizations, thus increasing the visibility on a higher level to discover, e.g., attack campaigns targeting a specific sector.

The incident clustering functionality has enabled security experts to observe common activities across multiple customer organizations, which yielded information used to fix repetitive false-positive detections. As such the method is primarily beneficial for cybersecurity service providers or large organizations who deal with a high number of incidents. Within the SAPPAN reference architecture, its functionality is best aligned with the recommendation system component.

Incident Response Automation. The evaluation of the analysis workflow automating typical incident response steps related to malware infection showed the advantages of linking shared data and automated processing. Although analysts typically use shared data, their manual querying during analysis significantly delays the entire incident resolution. Our proposed and evaluated automated processing appropriately shows how this problem can be overcome and nicely demonstrates the potential of this approach for other typical analyses performed in incident response. The evaluation also showed that automation is very good at reducing potential errors by analysts who may miss essential information during manual analysis. In addition, it also shows that results are consistent across analysts, and that the analysis times are similar with only small differences.

The malware analysis workflow appropriately demonstrates the potential associated with automation in threat detection and response. Primarily, it shows the strength of workload reduction (A.P1), where it efficiently implements common steps associated with malware analysis, as the analyst can upload a suspicious file and attend to other incidents while the file is being processed and analyzed. This is also related to the overall speeding up of the incident response (A.P2) as the individual analysis steps do not need to be performed manually, but it is sufficient to confirm the resulting incident response actions depending on the analysis result. Using a standardized procedure and available tools also offers easy monitoring of the entire analysis process and possible evaluation of all performed steps (A.P3). A vital role for the proper functioning of the workflow is the shared data based on which the analyzed sample is evaluated. Its quality (C.C5) significantly influences the processing results. Within the workflow, it is possible to use several different data sources and balance them against each other, which can significantly refine the analysis results (A.C4) and allow the analyst to trust the results of the automated processing.

The malware analysis workflow is part of the automation engine and is directly linked to the case management and dashboard in the SAPPAN reference architecture. At the beginning of the incident resolution, the analyst uses the integrating dashboard to upload a suspicious file according to the recommended steps. Subsequently, all the steps of the analytical workflow are executed, and the analyst only confirms the suggested actions within case management (e.g., blocking communication with malware-related domains on the firewall). Although not directly stated in Figure 2, the sharing system plays a key role, from which the malware analysis workflow takes the CTI and relevant IoCs, which the analyst can further act upon as part of the incident response follow-up.

Collaboration and Sharing. The results obtained from federated learning implemented over the MISP platform underscored its efficacy in collaborative model training while preserving data privacy and reducing FPR by training on a larger and more diverse dataset. Additionally, the shared models become the new form of IoCs in addition to the traditional forms such as hashes and IP addresses.

From this viewpoint, the sharing of models aligns effectively with the potential for preemptive defense (C.P1). The shared models encapsulate the detection of attacks of higher complexity or stealthiness than those typically described by the conventional forms of IoCs and the sharing itself enables other organizations to prepare the detection even before the attack hits them. At the same time, federated learning over the popular sharing platform offers the opportunity to involve multiple organizations to train models over more diverse and balanced datasets (C.P2). Additionally, the federated learning scheme offers the possibility to share metadata rather than the sensitive data itself, thus partially addressing the challenge of dealing with sensitive information (C.C1).

The validation of federated learning within the SAPPAN architecture centered on the DGA classifier use case. However, both the training process and the sharing of models remain agnostic to the specific use case. The successful experimentation involving federated learning across three organizations using MISP demonstrated the viability of this approach. A minor concern may be that all partial models exchanged during the training epochs remained stored in MISP. On the one hand, this may serve well to analyze the training phase, but on the other hand, it pollutes the MISP storage. Therefore, it may be practical to remove these partial models which requires each participating organization to remove its respective models after the training phase.

Data Sanitization. Our evaluation of different privacy-enhancing approaches in the binary DGA detection use case showed the difficulties of making sensitive information shareable. Data sanitization approaches that aim to make the data itself shareable and provide the receiving parties more flexibility in how the data can be utilized face the challenge of difficult-to-quantify privacy guarantees (as indicated by privacy evaluation of the FANCI feature extraction approach) or require finding a tradeoff between data protection and model utility (as indicated by the evaluation of the Bloom encoding approach). While alternative approaches such as SMPC and HE can be used to target a specific data processing task rather than the data itself, the evaluation of different available frameworks implementing SMPC and HE approaches revealed that the high communication cost introduced by these approaches renders their use infeasible for DGA detection in real-world settings, even after models have been simplified to reduce communication cost. Further model simplifications are expected to negatively impact the accuracy of the model. We found the identification of a suitable tradeoff between protection of samples and utility to be a labor-intensive process. This has multiple reasons:

(1) It is not always clear what a suitable level of protection is. As universal protection is commonly not achievable, the question of realistic threat models arises. The evaluation against specific threat models, however, further limits the general applicability of a solution, as seemingly small changes to the threat model may drastically weaken the protection provided. Hence, careful consideration may be required when applying a privacy-enhancing technology to a new use case or setting.
(2) Utility can be measured in various different ways. In the case of our Bloom encoding experiments, the utility was primarily determined by the impact on classification metrics. For the evaluation of cryptographic protocols, however, the utility was primarily impacted by the overhead caused by the protocols. This overhead could be reduced through model simplifications, which in turn impacted the quality of classification results. The evaluation of utility may thus require to consider additional tradeoffs, e.g., between computational overhead and classification results.
(3) It is difficult to determine how much utility can reasonably be sacrificed for the protection of data. Especially in practice, the perspectives of different roles may collide. On the one hand, ML engineers and incident handlers may prioritize the utility of developed tools to reduce the number of false positives. On the other hand, privacy engineers and data protection officers may prioritize the protection of data to minimize the

chance of negative consequences arising from collaboration. Finding a suitable tradeoff can hence also require negotiation between different roles in an organization.

(4) The process of identifying a suitable tradeoff between utility and protection of data for a specific use case depends on the chosen privacy-enhancing approach. For the three approaches we evaluated (Bloom encodings, FANCI features, and cryptographic protocols), the starting conditions for identifying such a tradeoff differed. For using the FANCI feature extractor as a sanitization tool, a good utility was given, but the question of its protective properties arose. In contrast, the cryptographic protocols provided strong mathematical guarantees and did in theory not negatively impact classification results, but utility was limited by the introduced overhead. For the Bloom encodings, neither the level of protection nor the provided utility was fixed, but had to be tuned via parameter values. Nevertheless, the question of whether a different approach could provide a better tradeoff in a specific use case always remains.

In terms of the identified challenges and potentials of collaboration, we found the primary challenge to be the choice of a suitable tradeoff between the risk of leaking sensitive information (C.C1) and the quality of sanitized data (C.C5). The secondary challenges were the universality of the trained models (C.C3), the machine-interpretability of data (C.C2), and the confidence in automated decisions (A.C4). Model universality was affected by the evaluated privacy-enhancing approaches, as they either utilize a specific data domain (Bloom encodings or FANCI feature vectors) or require different parties to take part in a specific SMPC/HE protocol. In regard to machine-interpretable data and automated processing, the choice of a suitable privacy-enhancing approach for a specific setting as well as the choice of respective parameter values requires human consideration. Potential future advancements through standardization efforts (C.P3) may help to mitigate this challenge through standardized data sanitization procedures for common data formats and use cases, providing practitioners with guidance in the selection of privacy-enhancing approaches and their parameter values. Even though none of the privacy-preserving approaches discussed as part of this work provided a perfect balance between privacy and data utility, we are optimistic that future improvements in respective technologies can leverage collaborative data generation and learning (C.P2), which was also indicated by our comparison between DGA detection classifiers trained in the local and the federated setting. We further see advances in the field of privacy-enhancing technologies contributing to preemptive defense (C.P1) by allowing organizations to share relevant data with a wider range of recipients in case they observe a novel kind of attack.

In the context of the SAPPAN architecture, the evaluation of different privacy-enhancing measures in the DGA detection use case indicates that a general and use case independent implementation of the data sanitization and transformation component may not be possible. Nevertheless, and due to its importance for activities related to sharing, a modular use case specific approach could be taken to implement this component. As such, independent tools could be developed, each of which focuses on the sanitization of specific data formats in specific use cases. It should, however, be considered that a combination of differently sanitized versions of the same dataset can leak information that cannot be retrieved when observing each sanitized version independently. We hence recommend keeping an internal record of which data has been shared in which form with other parties.

Playbook Management and Sharing. Our evaluation of the effective management and sharing of playbooks draws similarities from previous discussions on privacy-enhancing measures in data sharing, and explores the complexities and importance of playbook sharing through the SAPPAN architecture. One of the project's significant contributions is the development of a dedicated playbook management tool, the integration of a sharing object into the official MISP repository, and the verification of the toolkit, characterized by its compatibility with the CACAO standard. Its universality arises from a straightforward design including metadata that describes playbook properties and the playbook in its original format. The approach to formulating playbooks arrives in a semantic Web-based knowledge-capturing methodology. The flexibility inherent in the SAPPAN vocabulary, which allows for modeling confidentiality levels of individual resources, is an advancement in sharing. It facilitates refined access control and data sanitization to enable the dissemination of shareable playbooks.

Machine-readable playbook creation, management, and sharing open a new avenue in collaboration and automation potential. They especially address the potentials of response standardization (C.P3) and increased quality via human–machine teaming (A.P3), and may further help to reduce human workload (A.P1) and enable faster processing and resolution of incidents (A.P2) [72]. However, several challenges emerge in the context of playbook sharing and management. Firstly, maintaining confidentiality by addressing sensitive information (C.C1) is crucial in playbook management and sharing. Another significant challenge is the adaptability of the data model (C.C4). Playbooks can reveal sensitive information that is, for instance, related to the operating devices, network topology, and possible vulnerabilities of the sharing party systems. The introduction of privacy-preserving measures, such as data sanitization, addresses this challenge but also introduces complexities in maintaining playbook effectiveness. The balance between sharing comprehensive playbooks and maintaining confidentiality is a challenge that can be addressed by a dynamic data model such as the SAPPAN vocabulary based on semantic Web standards which offers flexibility in defining resources and proposing a confidentiality level for each resource. Secondly, the challenge of machine-interpretability (C.C2) of shared information is addressed by the machine-readability of playbook formats. The efforts on more effective cybersecurity response automation is an interesting topic for future work in this regard. Thirdly, the universality of playbook models (C.C3) raises questions about the adaptability of playbooks across diverse incident scenarios. Compatibility with well-known and widely used standards such as CACAO can address this concern and hint at where simplified processes could enhance collaborative efforts. Lastly, dealing with different nuances of an incident (A.C3), particularly in the context of supporting workflow branching in playbooks, is supported by both the CACAO and the SAPPAN format. While SAPPAN has made a significant effort in playbook management and sharing, it navigates a complex landscape of challenges and potentials. The efforts in aligning with established standards, developing a flexible vocabulary, and integrating sanitization and sharing mechanisms mark a step forward in collaborative cybersecurity [5]. However, the continuous evolution of cyber threats and organizational environments necessitates ongoing refinement and adaptation of these approaches.

Within the context of the SAPPAN reference architecture, the assessment of playbook management and sharing emphasizes the necessity for a user interface component dedicated to the creation, management, representation, and sharing of playbooks via the dashboard interface. A data sanitization module is critical to filter out confidential details from the playbooks and their respective steps prior to their distribution as CTI. It is also critical to establish various sharing levels, tailored for distinct dissemination scenarios and for parties with varying degrees of trustworthiness. Moreover, playbooks in disparate formats should be converted within the data transformation module to facilitate integration and enhance interoperability. Concerning automation efforts, it is advisable to identify low-impact incidents and integrate the relevant executable source code, which synchronizes with the organization's existing systems, into the playbook procedures. While full automation of response playbooks presents considerable challenges, partial or semi-automation of routine with low-impact tasks is promising for practical application. Moreover, linking the playbook management tool with the case management system is beneficial, not only to assist human operators and enhance recommendation algorithms, but also to collect feedback and insight to refine playbooks for subsequent incident management.

10 Conclusion

In this work, we have taken a practical look at the potentials and challenges that collaboration and automation may bring to a multi-purpose sharing infrastructure in the cybersecurity domain. As such, we started by identifying respective key potentials and challenges. To additionally provide a more technical context, we have proposed the SAPPAN reference architecture as a general set of technical components and their interactions that allow the execution of common tasks for sharing-enabled incident response. We then presented our work on the tasks of local attack detection, incident response, and sharing. Based on our validation results, we discussed the implications for the previously identified key potentials and challenges and the SAPPAN reference architecture.

Our results showed that both collaboration and automation have the potential to strengthen detection, analysis, and response capabilities of an organization. The consideration of ML models as IoCs allows to establish preemptive defense measures that can deal with novel incidents. The use of more diverse training datasets, e.g., via federated learning, further allows for creating better classifiers, which can increase the confidence in automated detection and reduce human workload due to lower FPRs. Collaboration in the security domain, however, has to deal with the matter of sensitive information, ranging from training data over model updates to playbooks. In our evaluation of respective approaches for the DGA detection use case, we saw that none of the three evaluated approaches poses a perfect solution. Handling sensitive information thus still requires human consideration and is expected to negatively affect the quality of data or restrict the universality of the shared data. When considering sharing response procedures as machine-readable playbooks, it is crucial to utilize an adaptable and universally applicable data model. The data model must be sufficiently flexible to cover different nuances of an incident within one playbook. In this context, the sharing of such playbooks would benefit from standardized and universally recognized playbook formats. Respective efforts have recently been made with the CACAO playbook format. Apart from that, the sharing of playbooks may further contribute to the standardization of common analysis and response procedures. In that regard, also the automation of such procedures should be considered, as seen by our implementation of the automated malware analysis workflow. The automation of steps commonly carried out for analyzing a malware sample can benefit human operators by reducing human workload, providing faster processing of the respective steps, and providing a detailed log of the entire analysis process. This benefits human–machine teaming without notable risk of disruption. Further, human operators can be assisted by incident similarity clustering, allowing to connect experience from incidents handled in the past to current incidents. By keeping a human in the loop and using automation procedures primarily to complement the work of human operators by carrying out common analysis tasks and suggesting response actions (rather than carrying them out directly), the risk of disruption is kept low.

While our validation results had implications for a majority of the identified key potentials and challenges, some of them have been discussed to a lesser extent. To provide some interesting directions for future work, we briefly discuss them in the following. In our work, we primarily focused on automating analysis tasks rather than response actions, since the latter has a greater potential for disruption. It further is associated with a higher implementation effort, as it needs to interact with different systems in an organization [72]. Further, the exploration of visualization techniques is of great interest, as it can assist in the assessment of data received from a collaborator, especially for complex forms of data such as trained deep learning models. The use of visualization techniques should also be considered in the context of automation of workflows, as it could increase confidence in automated decision making and benefit human–machine teaming through better monitoring. While we discussed problems that arise from sharing information that is potentially sensitive in an ML setting, we have not focused on problems arising from malicious collaborators. In this regard, methods for detecting data poisoning attacks are of significant interest to evaluate the quality of received data. Lastly, we also want to point out the importance of sanitization procedures for playbooks. Since cybersecurity playbooks can provide rich information to guide human operators in their work, it comes at the cost of an equally rich data format. Efforts to make playbooks shareable thus strongly rely on suitable sanitization procedures to ensure that organization-specific information is removed from a playbook that is prepared for sharing.

Acknowledgments

We would like to thank all members of the SAPPAN consortium, the project's advisory board, and its end-user committee. Our specials thanks go to Gabriela Aumayr, Franziska Becker, Benjamin Heitmann, Tomas Jirsik, Alexey Kirichenko, Karl-Heinz Krempels, Martin Laštovička, Ulrike Meyer, Christoph Müller, Josef Niedermeier, Mischa Obrecht, and Robert Rapp who made this work possible.

References

[1] Md Sahrom Abu, Siti Rahayu Selamat, Aswami Ariffin, and Robiah Yusof. 2018. Cyber threat intelligence—Issue and challenges. *Indonesian Journal of Electrical Engineering and Computer Science* 10, 1 (2018), 371–379.

[2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys* 51, 4, Article 79 (Jul. 2018), 35 pages. DOI: https://doi.org/10.1145/3214303

[3] Mehdi Akbari Gurabi, Avikarsha Mandal, Jan Popanda, Robert Rapp, and Stefan Decker. 2022. SASP: A semantic web-based approach for management of sharable cybersecurity playbooks. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 1–8.

[4] Mehdi Akbari Gurabi, Lasse Nitz, Andrej Bregar, Jan Popanda, Christian Siemers, Roman Matzutt, and Avikarsha Mandal. 2024. Requirements for playbook-assisted cyber incident response, reporting and automation. *Digital Threats* 5, 3, Article 34 (Oct. 2024), 11 pages. DOI: https://doi.org/10.1145/3688810

[5] Mehdi Akbari Gurabi, Lasse Nitz, Charukeshi Mayuresh Joglekar, and Avikarsha Mandal. 2024. Strengthening cyber defence through cooperative development and shared expertise in incident response playbooks. *ERCIM NEWS* 139 (2024), 44–46.

[6] Montdher Alabadi and Yuksel Celik. 2020. Anomaly detection for cyber-security based on convolution neural network : A survey. In *Proceedings of 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1–14. DOI: https://doi.org/10.1109/HORA49412.2020.9152899

[7] Bushra A. Alahmadi, Louise Axon, and Ivan Martinovic. 2022. 99% False positives: A qualitative study of SOC analysts' perspectives on security alarms. In *Proceedings of 31st USENIX Security Symposium (USENIX Security '22)*. USENIX Association, Boston, MA, 2783–2800. Retrieved from https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi

[8] Adham Albakri, Eerke Boiten, and Rogério De Lemos. 2019. Sharing cyber threat intelligence under the general data protection regulation. In *Privacy Technologies and Policy*. Maurizio Naldi, Giuseppe F. Italiano, Kai Rannenberg, Manel Medina, and Athena Bourka (Eds.), Springer International Publishing, Cham, 28–41.

[9] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials* 21, 2 (2019), 1851–1877. DOI: https://doi.org/10.1109/COMST.2019.2891891

[10] Roberto Andrade, Jenny Torres, and Susana Cadena. 2019. Cognitive security for incident management process. In *Proceedings of Information Technology and Systems (ICITS '19)*. Springer, 612–621.

[11] Frederik Armknecht, Youzhe Heng, and Rainer Schnell. 2023. Strengthening privacy-preserving record linkage using diffusion. *Proceedings on Privacy Enhancing Technologies* 2 (2023), 298–311.

[12] Tao Ban, Takeshi Takahashi, Samuel Ndichu, and Daisuke Inoue. 2023. Breaking alert fatigue: AI-assisted SIEM framework for effective incident response. *Applied Sciences* 13, 11 (2023), 6610.

[13] Vaclav Bartos, Martin Zadnik, Sheikh Mahbub Habib, and Emmanouil Vasilomanolakis. 2019. Network entity characterization and attack prediction. *Future Generation Computer Systems* 97 (2019), 674–686.

[14] Franziska Becker. 2021. *D3.9 Demonstrator of Visual Support for Designing Detection Models (Final Version)*. Technical Report. SAP-PAN Consortium. Retrieved from https://sappan-project.eu/wp-content/uploads/2022/11/D3.9-Demonstrator-of-Visual-Support-for-Designing-Detection-Models-Final-version-M30.pdf

[15] Franziska Becker, Arthur Drichel, Christoph Müller, and Thomas Ertl. 2020. Interpretable visualizations of deep neural networks for domain generation algorithm detection. In *Proceedings of 2020 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 25–29. DOI: https://doi.org/10.1109/VizSec51108.2020.00010

[16] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20)*. ACM, New York, NY, Article 14, 10 pages. DOI: https://doi.org/10.1145/3407023.3407045

[17] K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2016. Practical secure aggregation for federated learning on user-held data. In *Proceedings of NIPS Workshop on Private Multi-Party Machine Learning*, 5 pages. Retrieved from https://pmpml.github.io/PMPML16/papers/PMPML16_paper_8.pdf

[18] Xander Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and Michel van Eeten. 2020. A different cup of TI? The added value of commercial threat intelligence. In *Proceedings of 29th USENIX Security Symposium (USENIX Security '20)*. USENIX Association, 433–450. Retrieved from https://www.usenix.org/conference/usenixsecurity20/presentation/bouwman

[19] Xander Bouwman, Victor Le Pochat, Pawel Foremski, Tom Van Goethem, Carlos H. Ganan, Giovane C. M. Moura, Samaneh Tajalizadehkhoob, Wouter Joosen, and Michel van Eeten. 2022. Helping hands: Measuring the impact of a large threat intelligence sharing community. In *Proceedings of 31st USENIX Security Symposium (USENIX Security '22)*. USENIX Association, Boston, MA, 1149–1165. Retrieved from https://www.usenix.org/conference/usenixsecurity22/presentation/bouwman

[20] Juan Francisco Carías, Marcos R. S. Borges, Leire Labaka, Saioa Arrizabalaga, and Josune Hernantes. 2020. Systematic approach to cyber resilience operationalization in SMEs. *IEEE Access* 8 (2020), 174200–174221. DOI: https://doi.org/10.1109/ACCESS.2020.3026063

[21] Milan Cermak. 2020. *Stream-Based IP Flow Analysis*. Dissertation. Masaryk University. Retrieved from https://is.muni.cz/th/tgxb6/

[22] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys* 41, 3, Article 15 (Jul. 2009), 58 pages. DOI: https://doi.org/10.1145/1541880.1541882

[23] Peter Christen, Thilina Ranbaduge, Dinusha Vatsalan, and Rainer Schnell. 2019. Precise and fast cryptanalysis for Bloom filter based privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering* 31, 11 (2019), 2164–2177. DOI: https://doi.org/10.1109/TKDE.2018.2874004

[24] Peter Christen, Anushka Vidanage, Thilina Ranbaduge, and Rainer Schnell. 2018. Pattern-mining based cryptanalysis of Bloom filters for privacy-preserving record linkage. In *Advances in Knowledge Discovery and Data Mining*. Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi (Eds.), Springer International Publishing, Cham, 530–542.

[25] P. Cichonski, T. Millar, T. Grance, and K. Scarfone. 2012. *Computer Security Incident Handling Guide*, Vol. 800. NIST Special Publication, Gaithersburg, MD, 1–147 pages.

[26] Jason Creasey. 2013. *Cyber Security Incident Response Guide*. Crest.

[27] Cybersecurity and Infrastructure Security Agency. 2021. Cybersecurity Incident & Vulnerability Response Playbooks. Retrieved November 12, 2023 from https://www.cisa.gov/sites/default/files/publications/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf

[28] Morten Dahl, Jason Mancuso, Yann Dupis, Ben Decoste, Morgan Giraud, Ian Livingstone, Justin Patriquin, and Gavin Uhma. 2018. Private machine learning in TensorFlow using secure computation. arXiv:1810.08130. Retrieved from http://arxiv.org/abs/1810.08130

[29] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2020. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies* 4 (2020), 355–375.

[30] Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of ACM* 7, 3 (Mar. 1964), 171–176. DOI: https://doi.org/10.1145/363958.363994

[31] Arthur Drichel, Vincent Drury, Justus von Brandt, and Ulrike Meyer. 2021. Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES '21)*. ACM, New York, NY, Article 59, 12 pages. DOI: https://doi.org/10.1145/3465481.3470111

[32] Arthur Drichel, Mehdi Akbari Gurabi, Tim Amelung, and Ulrike Meyer. 2021. Towards privacy-preserving classification-as-a-service for DGA detection. In *Proceedings of 2021 18th International Conference on Privacy, Security and Trust (PST)*, 1–10. DOI: https://doi.org/10.1109/PST52912.2021.9647755

[33] Arthur Drichel, Benedikt Holmes, Justus von Brandt, and Ulrike Meyer. 2021. The more, the better: A study on collaborative machine learning for DGA detection. In *Proceedings of the 3rd Workshop on Cyber-Security Arms Race (CYSARM '21)*. ACM, New York, NY, 1–12. DOI: https://doi.org/10.1145/3474374.3486915

[34] Arthur Drichel, Ulrike Meyer, Samuel Schüppen, and Dominik Teubert. 2020. Analyzing the real-world applicability of DGA classifiers *(ARES '20)*. ACM, New York, NY, Article 15, 11 pages. DOI: https://doi.org/10.1145/3407023.3407030

[35] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*. Shai Halevi and Tal Rabin (Eds.), Springer, Berlin, 265–284.

[36] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407. DOI: https://doi.org/10.1561/0400000042

[37] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, 1054–1067. DOI: https://doi.org/10.1145/2660267.2660348

[38] European Union Agency for Cybersecurity (ENISA). 2023. *ENISA Threat Landscape 2023*. DOI: https://doi.org/10.2824/782573

[39] Marina Evangelou and Niall M. Adams. 2020. An anomaly detection framework for cyber-security data. *Computers & Security* 97 (2020), 101941. DOI: https://doi.org/10.1016/j.cose.2020.101941

[40] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFELearn: Secure aggregation for private FEderated learning. In *Proceedings of 2021 IEEE Security and Privacy Workshops (SPW)*, 56–62. DOI: https://doi.org/10.1109/SPW53761.2021.00017

[41] Gilberto Fernandes, Joel J. P. C. Rodrigues, Luiz Fernando Carvalho, Jalal F. Al-Muhtadi, and Mario Lemes Proença. 2018. A comprehensive survey on network anomaly detection. *Telecommunication Systems* 70, 3 (2018), 447–489. DOI: https://doi.org/10.1007/s11235-018-0475-8

[42] FIRST Standards Definitions and Usage Guidance. 2022. Information Exchange Policy (IEP). Retrieved December 6, 2023 from https://www.first.org/iep/

[43] FIRST Standards Definitions and Usage Guidance. 2022. Traffic Light Protocol (TLP)—Version 2.0. Retrieved December 6, 2023 from https://www.first.org/tlp/

[44] Inc Gartner. 2020. Security threat intelligence services reviews. Gartner. Retrieved November 12, 2023 from https://www.gartner.com/reviews/market/security-threat-intelligence-services

[45] Thomas Geras and Thomas Schreck. 2023. Sharing communities: The good, the bad, and the ugly. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, New York, NY, 2755–2769. DOI: https://doi.org/10.1145/3576915.3623144

[46] Daniel Gibert, Carles Mateu, and Jordi Planes. 2020. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications* 153 (2020), 102526. DOI: https://doi.org/10.1016/j.jnca.2019.102526

[47] Nitika Gupta, Issa Traore, and Paulo Magella Faria de Quinan. 2019. Automated event prioritization for security operation center using deep learning. In *Proceedings of 2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 5864–5872.

[48] Nicolas Heine. 2021. *Application Fingerprinting Based on System Events using Process Mining*. Master's thesis. RWTH Aachen University.

[49] Benedikt Holmes, Arthur Drichel, and Ulrike Meyer. 2021. Sharing FANCI features: A privacy analysis of feature extraction for DGA detection. In *Proceedings of 6th International Conference on Cyber-Technologies and Cyber-Systems (CYBER '21)*. IARIA XPS Press, 58–64.

[50] Martin Husák and Jaroslav Kašpar. 2018. Towards predicting cyber attacks using information exchange and data mining. In *Proceedings of 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 536–541.

[51] Martin Husák and Milan Čermák. 2022. SoK: Applications and challenges of using recommender systems in cybersecurity incident handling and response. In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*. ACM, New York, NY, Article 25, 10 pages. DOI: https://doi.org/10.1145/3538969.3538981

[52] Shouling Ji, Prateek Mittal, and Raheem Beyah. 2017. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys & Tutorials* 19, 2 (2017), 1305–1326. DOI: https://doi.org/10.1109/COMST.2016.2633620

[53] Tomas Jirsik and Petr Velan. 2021. Host behavior in computer network: One-year study. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 822–838. DOI: https://doi.org/10.1109/TNSM.2020.3036528

[54] Henrik Karlzen and Teodor Sommestad. 2023. Automatic incident response solutions: A review of proposed solutions' input and output. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23)*. ACM, New York, NY, Article 37, 9 pages. DOI: https://doi.org/10.1145/3600160.3605066

[55] Kathryn Knerler, Ingrid Parker, and Carson Zimmerman. 2022. *11 Strategies of a World-Class Cybersecurity Operations Center*. The MITRE Corporation.

[56] Dmitriy Komashinskiy and Andrew Patel. 2022. For security analysts, a picture may be worth more than a thousand words. Retrieved April 20, 2023 from https://sappan-project.eu/?p=1699

[57] Ifigeneia Lella, Eleni Tsekmezoglou, Marianthi Theocharidou, Erika Magonara, Apostolos Malatras, Rossen Svetozarov Naydenov, and Cosmin Ciobanu. 2023. *ENISA Threat Landscape 2023*. Technical Report. European Union Agency for Cybersecurity (ENISA). Retrieved from https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

[58] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-Closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of 2007 IEEE 23rd International Conference on Data Engineering*, 106–115. DOI: https://doi.org/10.1109/ICDE.2007.367856

[59] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2023. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2023), 3347–3366. DOI: https://doi.org/10.1109/TKDE.2021.3124599

[60] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2019. Reading the tea leaves: A comparative analysis of threat intelligence. In *Proceedings of 28th USENIX Security Symposium (USENIX Security '19)*. USENIX Association, Santa Clara, CA, 851–867. Retrieved from https://www.usenix.org/conference/usenixsecurity19/presentation/li

[61] Kun Liu and Evimaria Terzi. 2008. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM, New York, NY, 93–106. DOI: https://doi.org/10.1145/1376616.1376629

[62] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. L-Diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data* 1, 1, Article 3 (Mar. 2007), 52 pages. DOI: https://doi.org/10.1145/1217299.1217302

[63] Mandiant. 2013. OpenIOC 1.1. Retrieved December 6, 2023 from https://github.com/fireeye/OpenIOC_1.1

[64] Vasileios Mavroeidis, Pavel Eis, Martin Zadnik, Marco Caselli, and Bret Jordan. 2021. On the integration of course of action playbooks into shareable cyber threat intelligence. In *Proceedings of 2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2104–2108.

[65] Microsoft. 2023. *Microsoft Digital Defense Report 2023*. Retrieved from https://microsoft.com/mddr

[66] MISP. 2022. MISP Repository. Retrieved January 14, 2023 from https://github.com/MISP

[67] MITRE. 2012. CEE Language. Retrieved December 6, 2023 from https://cee.mitre.org/language/

[68] Pantaleone Nespoli, Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios kambourakis. 2017. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials* 20, 2 (2017), 1361–1396.

[69] Lasse Nitz, Mehdi Akbari Gurabi, Avikarsha Mandal, and Benjamin Heitmann. 2021. Towards privacy-preserving sharing of cyber threat intelligence for effective response and recovery. *ERCIM NEWS* 126 (2021), 33–34.

[70] Lasse Nitz and Avikarsha Mandal. 2023. DGA detection using similarity-preserving Bloom encodings. In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference (EICC '23)*. ACM, New York, NY, 116–120. DOI: https://doi.org/10.1145/3590777.3590795

[71] Lasse Nitz and Avikarsha Mandal. 2024. Bloom encodings in DGA detection: Improving machine learning privacy by building on privacy-preserving record linkage. *Journal of Universal Computer Science* 30, 9 (2024), 1224–1243. DOI: https://doi.org/10.3897/jucs.134762

[72] Lasse Nitz, Martin Zadnik, Mehdi Akbari Gurabi, Mischa Obrecht, and Avikarsha Mandal. 2022. From collaboration to automation: A proof of concept for improved incident response. *ERCIM NEWS* 129 (2022), 31–32.

[73] Don Norman. 2013. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.

[74] Stephen Northcutt. 2003. *Computer Security Incident Handling—Step by Step Guide*. SANS Institute.

[75] Megan Nyre-Yu, Kelly A. Sprehn, and Barrett S. Caldwell. 2019. Informing hybrid system design in cyber security incident response. In *Proceedings of 1st International Conference on HCI for Cybersecurity, Privacy and Trust (HCI-CPT '19) Held as Part of the 21st HCI International Conference (HCII '19)*. Springer, 325–338.

[76] OASIS. 2016. CybOX^{TM} Version 2.1.1. Retrieved December 19, 2024 from https://docs.oasis-open.org/cti/cybox/v2.1.1/cybox-v2.1.1-part01-overview.html

[77] OASIS. 2019. Open Command and Control (OpenC2) Language Specification Version 1.0. Retrieved December 6, 2023 from https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html

[78] OASIS. 2019. STIX Version 2.1. Retrieved December 6, 2023 from https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

[79] OASIS. 2022. CACAO Security Playbooks Version 1.1. Retrieved January 14, 2023 from https://docs.oasis-open.org/cacao/security-playbooks/v1.1/security-playbooks-v1.1.html

[80] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. 2006. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review* 36, 1 (Jan. 2006), 29–38. DOI: https://doi.org/10.1145/1111322.1111330

[81] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *Proceedings of the USENIX Security Symposium*. Thorsten Holz and Stefan Savage (Eds.), USENIX Association, 263–278.

[82] Davy Preuveneers and Wouter Joosen. 2021. Sharing machine learning models as indicators of compromise for cyber threat intelligence. *Journal of Cybersecurity and Privacy* 1, 1 (2021), 140–163. DOI: https://doi.org/10.3390/jcp1010008

[83] Théo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. arXiv:1811.04017. Retrieved from http://arxiv.org/abs/1811.04017

[84] D. Schlette, M. Caselli, and G. Pernul. 2021. A comparative study on cyber threat intelligence: The security incident response perspective. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2525–2556. DOI: https://doi.org/10.1109/COMST.2021.3117338

[85] Daniel Schlette, Philip Empl, Marco Caselli, Thomas Schreck, and Günther Pernul. 2023. Do you play it by the books? A study on incident response playbooks and influencing factors. In *Proceedings of 2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 60–60.

[86] Sandra Schmitz-Berndt. 2023. Defining the reporting threshold for a cybersecurity incident under the NIS directive and the NIS 2 directive. *Journal of Cybersecurity* 9, 1 (2023), 11 pages. DOI: https://doi.org/10.1093/cybsec/tyad009

[87] B. Schneier. 2014. The future of incident response. *IEEE Security & Privacy* 12, 5 (Sep./Oct. 2014), 96.

[88] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2009. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making* 9, 1 (2009), 41.

[89] Rainer Schnell and Christian Borgs. 2016. Randomized response and balanced Bloom filters for privacy preserving record linkage. In *Proceedings of 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 218–224. DOI: https://doi.org/10.1109/ICDMW.2016.0038

[90] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. 2018. FANCI: Feature-based automated NXDomain classification and intelligence. In *Proceedings of 27th USENIX Security Symposium (USENIX Security '18)*. USENIX Association, Baltimore, MD, 1165–1181. Retrieved from https://www.usenix.org/conference/usenixsecurity18/presentation/schuppen

[91] Sebastian Schäfer and Ulrike Meyer. 2022. A pipeline for DNS-based software fingerprinting. arXiv:2208.07042. Retrieved from https://arxiv.org/abs/2208.07042

[92] Avi Shaked, Yulia Cherdantseva, and Pete Burnap. 2022. Model-based incident response playbooks. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 1–7.

[93] Muhammad Sheeraz, Muhammad Arsalan Paracha, Mansoor Ul Haque, Muhammad Hanif Durad, Syed Muhammad Mohsin, Shahab S. Band, and Amir Mosavi. 2023. Effective security monitoring using efficient SIEM architecture. *Human-Centric Computing and Information Sciences* 13 (2023), 1–18.

[94] Jonathan M. Spring and Phyllis Illari. 2021. Review of human decision-making during computer security incident analysis. *Digital Threats: Research and Practice* 2, 2 (2021), 1–47.

[95] Latanya Sweeney. 2002. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[96] Tines. 2022. *Voice of the SOC Analyst*. Retrieved April 4, 2024 from https://www.tines.com/reports/voice-of-the-soc-analyst

[97] Daniel Tovarnak and Martin Lastovicka. 2022. Malware Pipeline. Retrieved September 1, 2023 from https://github.com/CSIRT-MU/csirtmu-sappan-malware-evaluator

[98] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the 3rd ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20)*. ACM, New York, NY, 61–66. DOI: https://doi.org/10.1145/3378679.3394533

[99] MuhammadFahad Umer, Muhammad Sher, and Yaxin Bi. 2017. Flow-based intrusion detection: Techniques and challenges. *Computers & Security* 70 (Sep. 2017), 238–254. DOI: https://doi.org/10.1016/j.cose.2017.05.009

[100] Anushka Vidanage, Thilina Ranbaduge, Peter Christen, and Rainer Schnell. 2019. Efficient pattern mining based cryptanalysis for privacy-preserving record linkage. In *Proceedings of 2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 1698–1701. DOI: https://doi.org/10.1109/ICDE.2019.00176

[101] Manfred Vielberth, Fabian Böhm, Ines Fichtinger, and Günther Pernul. 2020. Security operations center: A systematic study and open challenges. *IEEE Access* 8 (2020), 227756–227779.

[102] Thomas D. Wagner, Khaled Mahbub, Esther Palomar, and Ali E. Abdallah. 2019. Cyber threat intelligence sharing: Survey and research directions. *Computers & Security* 87 (2019), 101589.

[103] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469. DOI: https://doi.org/10.1109/TIFS.2020.2988575

[104] Jun Xu, Jinliang Fan, M. H. Ammar, and S. B. Moon. 2002. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of 10th IEEE International Conference on Network Protocols*, 280–289. DOI: https://doi.org/10.1109/ICNP.2002.1181415

[105] ytisf. 2023. theZoo—A Live Malware Repository. Retrieved October 19, 2023 from https://thezoo.morirt.com/

[106] Mingchun Yun, Yuqing Lan, and Tao Han. 2017. Automate incident management by decision-making model. In *Proceedings of 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, 217–222.

[107] Martin Zadnik. 2021. Sharing response handling information. Retrieved December 19, 2024 from https://sappan-project.eu/wp-content/uploads/2022/11/D5.8-Sharing-Response-Handling-Information-M30.pdf

[108] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu an Tan. 2019. Secure multi-party computation: Theory, practice and applications. *Information Sciences* 476 (2019), 357–372. DOI: https://doi.org/10.1016/j.ins.2018.10.024

[109] Bin Zhou, Jian Pei, and WoShun Luk. 2008. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations Newsletter* 10, 2 (Dec. 2008), 12–22. DOI: https://doi.org/10.1145/1540276.1540279

[110] Athanasios Zigomitros, Fran Casino, Agusti Solanas, and Constantinos Patsakis. 2020. A survey on privacy properties for data publishing of relational data. *IEEE Access* 8 (2020), 51071–51099. DOI: https://doi.org/10.1109/ACCESS.2020.2980235

Received 30 December 2023; revised 29 June 2024; accepted 12 November 2024

Building a Better SOC: Towards the Ontology for Security Operations Center Assistance and Replication (OSCAR)

JUSTIN NOVAK, ANGEL HUECA, CHRISTOPHER RODMAN, and SAMUEL PERL, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
TRAVIS BREAUX, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
JUSTIN VALDENGO, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

There are many methods for developing a Security Operations Center (SOC) or SOC capability. However, there currently exists no unified approach which comprehensively outlines the people, processes, and technology required for developing a SOC, or how an organization might implement those into an effective SOC capability. This article outlines a data gathering process used to compile knowledge necessary for a proposed Ontology for SOC Creation Assistance and Replication, which can serve as a solution to the gap in the current body of knowledge. An ontology such as the one proposed here would leverage the collective experience of a large cadre of cybersecurity experts with deep knowledge in fields related to Security Operations and the development of SOCs. Using interview methods and analysis, the knowledge of how these experts approach the problem of creating new SOC capabilities within a set of known constraints can be captured and codified. The result is a comprehensive body of structured knowledge outlining what critical decisions are made during the process, and how those decisions affect the implementation of People, Processes, and Technology which become part of a SOC. It is this body of knowledge which can be organized and presented as a formal ontology.

CCS Concepts: • **Security and privacy → Formal security models**; **Usability in security and privacy**; **Intrusion detection systems**;

Additional Key Words and Phrases: SOC, Security Operations, Security Operations Center, People, Process, Technology, Incident Response, Ontology

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
Authors' Contact Information: Justin Novak (corresponding author), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: jnovak@cert.org; Angel Hueca, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: alhueca@cert.org; Christopher Rodman, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: cirodman@cert.org; Samuel Perl, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: sjperl@cert.org; Travis Breaux, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: tdbreaux@andrew.cmu.edu; Justin Valdengo, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: jvaldengo@cert.org.

ACM 2576-5337/2025/3-ART6
https://doi.org/10.1145/3722233

ACM Reference format:
Justin Novak, Angel Hueca, Christopher Rodman, Samuel Perl, Travis Breaux, and Justin Valdengo. 2025. Building a Better SOC: Towards the Ontology for Security Operations Center Assistance and Replication (OSCAR). *Digit. Threat. Res. Pract.* 6, 1, Article 6 (March 2025), 22 pages.
https://doi.org/10.1145/3722233

1 Introduction

With cyber-attacks becoming more prevalent and sophisticated, organizations face increased challenges in effective cybersecurity incident response. "By definition, a **Security Operations Center (SOC)**, bundles organizational security roles, essential security services, and tools [Schlette et al., 2021]." SOC related roles, services, and tools are often mandated by specific regulatory requirements or may be needed to meet conditions for alignment with standards such as the Identify, Protect, Detect, Respond, and Recover capabilities outlined in the **National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF)**. Other security frameworks, such as the **International Organization for Standardization (ISO)** 27001 have similar requirements. The complexities and cost of meeting such requirements or complying with a specific security control framework can be overwhelming and discouraging for many organizations developing and implementing a SOC. In this article, we examine how gathering and codifying the knowledge required to carry out SOC development and implementation tasks may improve outcomes in this area. This research outlines a qualitative process for collecting and analyzing data related to SOC development. Using this process, we build a unique dataset which comprehensively describes how SOCs are developed in a defined set of circumstances. Finally, we propose that by formalizing this data it may be organized into an ontology—a structured body of knowledge. The research team has been involved in SOC creation and development for several years, and that experience in turn builds upon the decades of work in developing other types of incident response and cybersecurity capabilities, such as **Computer Security Incident Response Teams (CSIRTs)**, etc.[1]

The research gap we address here is the incomplete understanding of how organizations develop effective SOC capabilities, once the decisions to develop a SOC has been made. We do so by creating a new dataset which offers insights into the process and suggests the creation of a formal ontology for the field. This new data and knowledge help to close the existing gap in research and practice.

1.1 Related Work

Within an organization, "a SOC is an organizational unit operating at the heart of all security operations" [Vielberth et al., 2020]. A SOC is the principal defense group within an organization, typically focusing on capabilities such as monitoring, preventing, responding, and reporting [Kokulu et al., 2019]. SOCs are also multifaceted and complex structures that enhance the organization's security posture [Vielberth et al., 2020]. In order to be effective, the SOC must have defined processes and procedures for managing security incidents, as well as have its performance measured using established metrics [Mughal, 2022].

Torres [2015] identifies the "Triad of Security Operations: People, Process, and Technology," central to security operations (SecOps). Moreover, the author illustrates the need for SecOps to align with organizational culture, along with the integration of "multiple functions (people), varying processes and procedures (process) (**People, Process, Technology (PPT)**), and disparate security products (technology)," as seen in Figure 1.

Importantly, not all process frameworks focus exclusively on the PPT criteria. The **Information Technology Infrastructure Library (ITIL)** process operating model version 3, for example refers to people, process, products, and partners as the core set of practices for any IT activity or service. However, the ITIL framework is generalized

[1]In this study we use both the term "CSIRT" and the term "SOC." While there are many definitions of each term, we generally adhere to the definitions used by the Forum of Incident Response and Security Teams (FIRST) which are available in many FIRST publications, including the *CSIRT Services Framework* [FIRST, 2023] and the *Team Types within the Context of Services Frameworks* [FIRST, 2024].

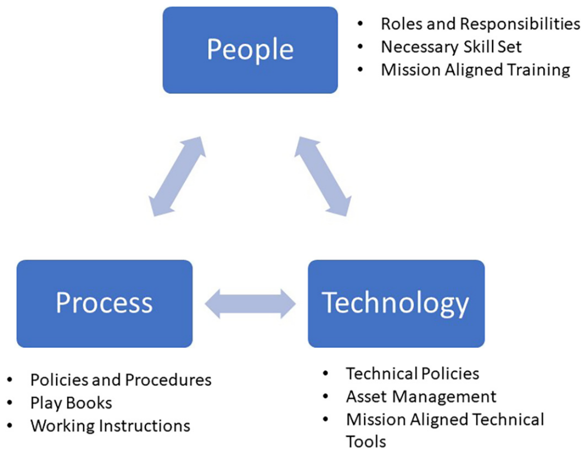


Fig. 1. PPT.

for all IT services, and not specific to security functions, and later versions, such as ITIL version 4 have moved toward a different framing of how services are implemented [Susnjara and Smalley, 2024]. For this reason, we focus here on the PPT framework common to IT security practices.

Due to the complex nature of the SOC, and the interdependencies of PPT, many organizations struggle with successful SOC implementation. SOCs can perform a variety of services based on regulatory requirements or organizational needs and mission. Issues begin to arise when the synergy between PPT begins to break down. This can be attributed to several factors. Policy and process drive how people and technology interact, how people should use technologies within the organization, and what services those technologies should support.

Components of a SOC should be determined once the organization's unique cybersecurity risks have been identified and aligned with any specific regulatory requirements. Additionally, services offered by the SOC should support an organization's mission and assist an organization in monitoring, detecting and responding to cyber threats. SOC services are an integral part of maintaining an organization's cybersecurity posture. The **Forum of Incident Response and Security Teams (FIRST)** documents in the *Team Types within the Context of Services Frameworks* guide which services are essential to different security team types. For SOCs, Information Security Event Management and Information Security Incident Management are noted as required services for SOC teams. As a result, this study focuses on these two SOC functions, which are alternatively referred to here as Network Monitoring and Incident Response.

One common issue in SOC development occurs when an organization has a low level of understanding of what the SOC could be capable or how to align the SOC functions to meet organizational needs and mission. Since security is not a "one-size fits all," stakeholders often find themselves puzzled when it is explained that the implementation of a technical solution will not solve their security issues. The importance and role of the SOC lie in its support of the core tenets of cybersecurity, that of information **Confidentiality, Integrity, and Availability (CIA)**, otherwise known as the CIA triad. The success of implementing information CIA within an organization relies heavily on the previously mentioned PPT. As observed by the real-life interactions of the research team with several stakeholders, PPT do not always work seamlessly together, consequently leading to the concept for this research project.

SOC failure is often attributed to "a mix of technical and human centric issues [Kokulu et al., 2019]." As noted by Torres [2015], SOCs need to have systemic communication and collaboration in order for the SOC to function effectively. In addition to SOC issues centered on PPT, Vielberth et al. [2020] also identified "integration of domain knowledge" as a challenge. This type of challenge arises from the ever-changing attack surface, with

Table 1. SOC Challenges [Kokulu, 2019]

Category	**Subcategory**
Operational Issues	Low visibility on devices and network topology Inferior defense against specific types of attacks Slow response speed of the SOC Inefficient evaluation metrics Insufficient budget
Technological Issues	Overloaded and low-quality threat intelligence Poor quality of reports and logs High false-positive rate Malfunctioning of SOC tools Insufficient automation level of SOC components Poor usability of SOC systems Challenge of scaling SOC technologies
Human Knowledge Issues	Low situational awareness Insufficient analyst training

people and tools unable to detect threats not defined by existing rules or mitigations. Through 18 semi-structured interviews of SOC analysts and managers, a 2019 study conducted by Kokulu et al. dove deeper into SOC operation challenges. It is important to highlight SOC challenges for the purposes of this study, as the process and body of knowledge described here aims to address SOC development challenges through the creation of an ontology. The challenges uncovered by Kokulu et al. [2019] are presented in Table 1 below.

Further exploration of qualitative research literature demonstrates a common acceptance of the SOC PPT model. As noted by Teoh et al. [2019], lack of workforce skills in people, lack of an implementation plan and budgeting process, along with rapid technology acceleration were found to be key challenges for a SOC.

1.1.1 Research Gaps and Motivation. There exist several related modern assessment frameworks which use capability and maturity criteria to evaluate a SOC. For example, the SOC CMM framework looks at 5 domains and 25 "elements" of SOC development. The key SOC CMM domains are PPT, business, and services [Van Os, 2023]. Other models, such as SIM3, define quantitative measurements to calculate success factors for security team implementations [Stikvoort, 2015], though SIM3 specifically targets development of national CSIRT teams. Although National CSIRT team responsibilities tend to differ from an internal organizational SOC as described in the CSIRT Services Framework [FIRST—Forum of Incident Response and Security Teams, 2023], there are many commonalities between people development, tools, and processes. The output of both SOC CMM and SIM3 are quantitative scores based on key aspects of PPT maturity and serve as a measure for benchmarking performance. These frameworks are helpful for developing a SOC or SOC capability. However, there remains a gap between these frameworks—which are designed for assessing a SOC—and a comprehensive, full body of knowledge regarding SOC development. Such as a knowledge base or ontology is described in this study as a means of addressing that gap. In short, while frameworks can be useful tool for thinking about or approaching a problem in a systematic way, they lack a complete and comprehensive description of the domain, including details and particular information necessary for a comprehensive understanding of the entire domain.

It is clear that there is a need for an a more formal knowledge body such as an ontology to assist organizations in not only meeting SOC development challenges but, also in determining the appropriate technologies and capabilities needed to instantiate a fully functional SOC.

The knowledge collected and classified in this study ultimately proposes the **Ontology for SOC Creation Assistance and Replication (OSCAR)** which aims to alleviate SOC development and implementation concerns by capturing and organizing information related to the PPT required for a SOC. It also aims to provide a method for aligning this knowledge with organizational requirements provided by an organization.

2 Methods

To establish a clear understanding of the challenges analysts and managers experience in the SOC environment, this research utilizes a data gathering approach for building a deep body of knowledge. To better understand SOC development processes, data gathering has focused on gaining a better understanding of three key areas highlighted in the literature: PPT. As noted by Tenny et al. [2022], qualitative research explores and provides deeper insights into real-world problems, meaning that this approach is ideal for a data gathering study such as the one described here.

Qualitative research refers to research that produces findings through means not related to statistical analysis or other methods of quantification [Cypress, 2015]. Qualitative research is used when researchers intend to understand the human experience; this may be related to individual behaviors, organizational functions, interactional relationships [Cypress, 2015; Donalek, 2005], or as in this study, SOCs. As expressed by Trivedi and Chan [2023], "qualitative research is primarily exploratory in nature and helps the research better understand motivations, needs, processes, and rationale for behaviors." For the purposes of this study, the researchers aim to understand the challenges that organizations face in standing up a SOC, from the perspective of those related to creation and operations, senior leadership, managers, and analysts.

Further exploration of the literature revealed that qualitative research is an inquiry process of understanding that explores a social or human problem ; [Khan, 2014]. This inquiry process is typically conducted in the form of an unstructured interview, as the interview is "by far the most common method of qualitative data collection" [Donalek, 2005]. As noted by Donalek [], interview questions can be semi-structured or unstructured, based on the purpose of the study, with an unstructured interview used to draw out participant experiences.

For this study, the research team interviewed 24 cybersecurity professionals ranging from senior leadership to analysts, and varying sectors from government and private industry. Interviews began with a series of questions designed to establish individual level of expertise and sectors (to include government) in which participants have worked. Once each participant's area or sector was identified, a series of interview questions based on a specific scenario was presented. The following sections describe the interview and information gathering process in more detail, along with the results and some brief discussion.

2.1 Research Design

This study describes the collection and organization of interview data combined with experience which can be used to form a formal knowledge base. The data and rules within the knowledge base (to date) are based on knowledge and experience of security experts with a history of assessing and building SOC teams.

To develop a more comprehensive knowledge base (beyond a framework alone), this research collected data in four phases: (1) Data Collection—design and conduct interviews with security experts; (2) Data Transformation—coding the interview transcripts to identify SOC development concepts; (3) Data Analysis—analyze the codes to identify common patterns; and (4) Results—building those concepts and patterns in to a structured body of knowledge. Data is primarily obtained by asking scenario-based questions of experts interviewed by the research team. This scenario-driven approach aligns with common knowledge elicitation approaches used in research, such as **Scenario Based Requirements Elicitation (SBRE)** as proposed by Holbrook [1990]. In SBRE, a scenario is used by designers of a system to understand what requirements the users of that system may have. Scenario selection and development is discussed in further detail in Section 2.2 below.

The interview structure and modality used in this study is that of a semi structured conversation with each expert. This structure allowed each expert to elaborate on their perspectives of building a SOC, and to keep interviews within the structured focal areas of PPT without restricting thought processes of the expert interviewee. This informal interview approach was chosen primarily to achieve insight, experiences, and understandings from experts [Rowley, 2012]. This also differentiates the research by uncovering new success factors that are not considered by current quantitative SOC evaluation methodologies.

Identifying security operations **Subject Matter Experts (SMEs)** and securing interviews with those individuals was primarily done by arranging voluntary in-person meetings during the FIRST annual conference held in Montreal Canada, in June 2023. The annual conference draws a wide array of cybersecurity expertise from a variety of international organizations. Participants of the conference often hold roles such as Chief Information Security Officer, Cybersecurity Researcher, SOC Manager, and includes members of national CSIRTs, Product Security and Incident Response Teams as well as other members of government, academia, and private industry. Solicitation of volunteers to participate in research interviews at this forum ensured a variety of perspectives and experiences. Further targeting security operations experience, additional calls for participation in the study were done through a local chapter of the International Information System Security Certification Consortium. Specific background information regarding the expert population is shared in further detail in Section 3 below.

Utilizing these various solicitation methods and forums allowed for us to selectivity choose the experience and expertise thresholds for interview candidates and helped to ensure that interview participants were truly cybersecurity operations SMEs. Interview eligibility was also based on potential participants' current job role. Qualifying job roles for the study were determined using the framework set forth by the **NIST Workforce Framework for Cybersecurity (NICE)** [Petersen et al., 2020]. While not all candidate job role titles exactly matched the specialty area keywords identified in the NICE framework, our research team was able to determine which responsibilities were similar in nature and meaning. The diversity of participants for this qualitative study also allowed for the collection of alternate perspectives on building a SOC, such as from SMEs in closely related fields including network operations or IT administration. Other expertise-defining factors such as years of cybersecurity experience, education, and certification background were collected using a demographic questionnaire during each interview. To meet the objectives of building a dataset specific to SOC development needs, the candidates' qualifying duties were recorded and mapped to Specialty Areas of interest within the NICE framework. Table 2 below describes the qualifying duties and their mapping to equivalent NICE functions and Specialty Areas.

During the data collection interviews, we preserved participant anonymity and privacy to ensure that candid conversations could take place for each interview. This was accomplished first using a confidentiality agreement and second by avoiding use of interviewee names or organizational information during the interview or as each interview was recorded and transcribed. Instead, interviewees were asked to describe their knowledge in the context of fictional scenarios. The use of such scenarios adds to the robustness of the data collection by encouraging recall and allowing users to expand beyond their current working situation, while also providing anonymity, as described below. Additionally, interviews took place in either a private meeting space or over a secured virtual meeting platform. Finally, any names, organizational affiliations, or identifying information were redacted from the interview dataset.

The determination of sample size for this research focused on ensuring that enough SOC knowledge would be gathered to support the development of a detailed and rich set of codes. The key attributes interview data are their duration, and the number of interviews conducted—both of which affect the total amount of data gathered. The time required for each interview is of particular importance as a planning consideration, as the need to gather a sufficient volume of data and expert insights is directly correlated to the amount of discussion which can be capture. In planning, anticipated timeframes for each interview fell between 30 and 60 minutes in length. In total, we completed 24 interviews each with and an average length of 40+ minutes. This exceeded our original data gathering goal.

Table 2. Qualifying Duty NICE Framework Mapping

Qualifying Duty	NICE Function	NICE Specialty Area
Building or Establishing SOCs and Cybersecurity Programs	Oversee and Govern	Strategic Planning and Policy, Executive Cyber Leadership
Managing or Leading SOCs	Collect and Operate, Oversee and Govern	Cyber Operations Planning, Cybersecurity Management
Researching SOCs and Their Functions	Securely Provision	Software Development, Systems Development, Technology R&D
Consulting for SOCs	Protect and Defend, Securely Provision	Cyber Defense Analysis, Risk Management, Systems Architecture, Systems Requirements Planning, Technology R&D
Working in SOCs within Senior Capacity	Investigate, Operate and Maintain, Protect, and Defend	Cyber Investigation, Incident Response, Cyber Defense Infrastructure Support

The first step as described above involves the individual interviews from the cohort of cybersecurity operations SMEs. Secondly, each recorded interview was transcribed into a text document. The third step in the research method was to analyze the transcripts and map key terms and phrases from interviewee answers to a set of codes. Our goal was to map common terms and themes as described by each of the experts. Codifying answers will allow for the construction of a knowledge base as a next step through the use of grounded theory [Noble and Mitchell, 2016].

2.2 Interview Scenario Development

Two key components of developing a knowledge base (particularly one used to develop a future ontology) are fidelity of data and verification that related questions reflect unbiased perspective on aspects of the subject [Rowley, 2012] To satisfy these knowledge base development components specific to the subject of SOC creation and development, we used a scenario-based approach (rather than asking interviewees about their own organizational security operations programs free form). This approach facilitated openness from the experts who may have otherwise been reluctant to provide specific details or complete insights due to the sensitive nature of cybersecurity operations for most organizations. Overall, 18 different scenarios were developed as options for use in any single interview. Each scenario was differentiated by organization size, organization sector, and primary SOC motivation. A breakdown of each scenario can be found in Table 3. To select a scenario for each expert, we first asked an initial set of background questions about the interviewees experience. This enabled the research team to select a scenario to best match interviewee skill set and organizational experience.

Background questioning also allowed the interviewers to validate the expertise of the person being interviewed. These questions focused on areas such as educational and certification credentials that were applicable to cybersecurity fields of study. Furthermore, these conversations also helped to discover task domains of each interviewee's present role, allowing for additional validation of expertise even in cases where opinions disagreed throughout the interview data gathering process, a key validation method of expertise as described by Shanteau [2015]. SOC implementation scenarios included a range of possible scenarios, allowing interviewers to tailor each interview to be specific to the selected organization sector and size. For example, a small private organization may not be subject to regulatory requirements for implementing a SOC, whereas a large government organization may be bound by specific criteria to establish a SOC.

Table 3. Different Interview Scenarios Used during Data Collection

No.	Organization Size	Organization Sector	Motivation
1	Small	Government	Regulatory Requirement
2	Small	Government	Response to an Incident
3	Medium	Government	Regulatory Requirement
4	Medium	Government	Response to an Incident
5	Large	Government	Regulatory Requirement
6	Large	Government	Response to an Incident
7	Small	Private Industry	Regulatory Requirement
8	Small	Private Industry	Response to an Incident
9	Medium	Private Industry	Regulatory Requirement
10	Medium	Private Industry	Response to an Incident
11	Large	Private Industry	Regulatory Requirement
12	Large	Private Industry	Response to an Incident
13	Small	Critical Infrastructure	Regulatory Requirement
14	Small	Critical Infrastructure	Response to an Incident
15	Medium	Critical Infrastructure	Regulatory Requirement
16	Medium	Critical Infrastructure	Response to an Incident
17	Large	Critical Infrastructure	Regulatory Requirement
18	Large	Critical Infrastructure	Response to an Incident

Finally, each scenario was restricted to only address the network monitoring and incident response duties and functions of the SOC. This restriction came after considering all possible SOC duties such as “vulnerability discovery/research, artifact and forensic evidence analysis, and awareness building” identified as “Service Areas” by the CSIRT Services Framework [CSIRT Services Framework Version 2.1, n.d.] as noted above. The functions selected are considered core SOC tasks, meaning they carry the most weight for SOC development. Addressing all various functionalities such as those listed in the CSIRT Services Framework would be prohibitively time consuming and is an area for future research. Addressing a limited set of SOC functionality also allowed the interview team to focus on the depth of knowledge required for these limited functions rather than the breadth of functionality and duties that a SOC could undertake, minimizing overcomplexity in the interview discussions.

2.2.1 Interview Questions. Interview questions were crafted around core pillars of a SOC—PPT—with questions to address each pillar. Interviews were conducted in this manner to encourage interviewees to provide description and justification for each of the primary areas of SOC development. Emphasis was placed on developing questions to help understand processes relevant to establishing the PPT for building a SOC within the prescribed scenario. Themes for the questions are shown in Table 4 below as *exploratory factors*. However, the semi-structured format of the interviews allowed for flexibility when an expert interviewee provided insight outside of these areas.

People Questions. Addressing people aspects of SOC development is based on a determination of knowledge and skills required for the duties that are expected of the SOC team. Institutionalized frameworks, such as NIST special publication 800-181 revision 1, define these duties as knowledge and skills which are used for fundamentally describing qualifications for different job role functions within cybersecurity [Petersen et al., 2020]. Frameworks such as the Workforce Framework for Cybersecurity (NICE Framework) describe the knowledge and skills required to perform tasks within the cybersecurity profession. Aimed at understanding the skill sets that SOC professionals deemed indispensable to the practical operation of the SOC, questions presented to interviewees were set forth to prioritize known skills and to discover SOC team attributes that lead to the successful operations of the team.

Table 4. Summary of Interview Questions

Question Percentage	Interview Component	Exploratory Examples
16.6%	Interviewee Demographics	Years of experience, education, and certification Organizational metrics (assets, budget, people)
33.3%	Motivations/Challenges	Goals/Objectives and evaluations Funding, staffing, tools acquisition
16.6%	Technology	Tool needs identification Third-party integration
16.6%	People/Process	Policies, standards, and procedures People metrics, skills, and abilities
16.6%	Prioritizations	Organizational risk

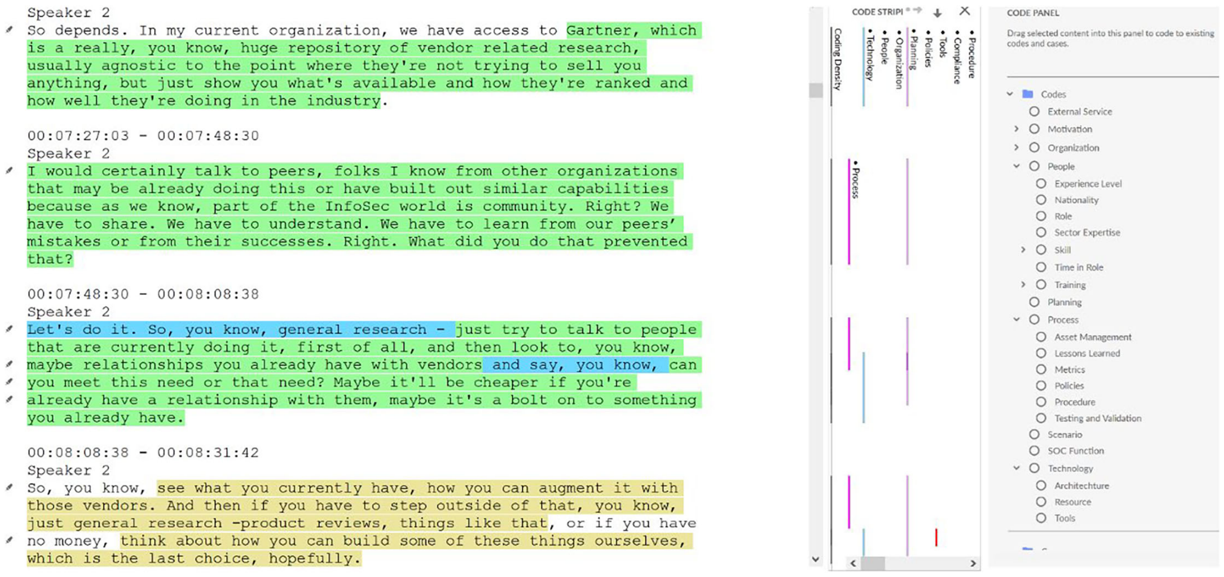
Interview questions also sought to determine the number of SOC personnel needed in each of the scenarios. The purpose of doing so was to understand the motivating factors that drive decisions about the number of personnel employed to staff the SOC and fill the varying SOC roles that interviewees felt were of importance. Data collected from the people-related questions were used to determine what interviewees felt made up the ideal composition of the SOC team from a practical perspective. As with the overall data collection approach, the duties or functions of the SOC were limited in each scenario to only account for network monitoring and incident response, which are considered core functions of most SOCs.

Process Questions. Process questions address the identification of policies, standards, and procedures that are required for a SOC to operate. These questions not only aim to identify and prioritize documents to support a SOC but also to address challenges with the development and sustainability of these documents. Some examples of process challenges and solutions have been identified by Onwubiko and Ouazzane [2022] and Mansfield-Devine [2016] where processes are identified as key components of sustainability and success for the SOC. In this spirit, interview questions for this study asked experts to identify governing articles such as charter or concept of operations policies that form strategy for the SOC and then to elaborate on the establishment and application of these articles.

Technology Questions. Technology-based questions were designed to discover the appropriate hardware and software that the interviewee felt needed to be implemented for the SOC to function successfully. Technological aspects include what tools the SOC would need to operate based on the functions that they are asked to perform within the interview scenario. Technology considerations related to additional SOC functions beyond the core functions were not discounted by the interview team but were included in the data gathering. This is mainly because the SME may have felt an important need for them to be part of SOC operations. Examples include tools used for vulnerability management, threat intelligence, or security engineering. Additionally, technology questions also meant to uncover the motivating factors, requirements, priorities, and challenges involved with implementing hardware and software tools for a SOC to utilize.

2.3 Data Coding

After the interviews were completed, each interview was carefully transcribed into text files. Utilizing grounded theory methods similar to those identified by Noble and Mitchell [2016], the transcribed interviews were analyzed to develop "categories and analytic codes" from the data. The intent of this process of coding each transcript is to identify common themes (which become data "codes") in phrases used by interviewees, relationships between different themes, and attributes which describe each theme. This is similar to the approach used by Grishman et al. [2002], which extracted from text information to detect disease outbreak, as well as specific keywords and



```
Speaker 2
So depends. In my current organization, we have access to Gartner, which
is a really, you know, huge repository of vendor related research,
usually agnostic to the point where they're not trying to sell you
anything, but just show you what's available and how they're ranked and
how well they're doing in the industry.

00:07:27:03 - 00:07:48:30
Speaker 2
I would certainly talk to peers, folks I know from other organizations
that may be already doing this or have built out similar capabilities
because as we know, part of the InfoSec world is community. Right? We
have to share. We have to understand. We have to learn from our peers'
mistakes or from their successes. Right. What did you do that prevented
that?

00:07:48:30 - 00:08:08:38
Speaker 2
Let's do it. So, you know, general research - just try to talk to people
that are currently doing it, first of all, and then look to, you know,
maybe relationships you already have with vendors and say, you know, can
you meet this need or that need? Maybe it'll be cheaper if you're
already have a relationship with them, maybe it's a bolt on to something
you already have.

00:08:08:38 - 00:08:31:42
Speaker 2
So, you know, see what you currently have, how you can augment it with
those vendors. And then if you have to step outside of that, you know,
just general research -product reviews, things like that, or if you have
no money, think about how you can build some of these things ourselves,
which is the last choice, hopefully.
```

Fig. 2. Transcript coding.

phrases, before classifying each example from the relevant transcript. In the case of Grishman et al. [2002], specific keywords related to the creation of a classification system used against collections of infectious disease outbreaks to form a database. The resultant dataset was then used to sort, filter, and search classifications or codes. Data collected from the interview transcripts defined practical characteristics, perspectives, and motivations similar to those expressed by Trivedi and Chan [2023].

As in the Grishman study, SOC knowledge for this study was harvested using an approach of analyzing transcription data, identifying and classifying keywords, phrases, and passages. and then creating classification codes. Coding classification was conducted using multiple passes through the transcript. In the first pass, initial high-level categories were identified and coded based on the SME's answers to the scenario-based questioning during each interview. The second pass on the same transcript revealed further sub-categorizations of phrases and passages. Codes generated on the second and any subsequent reviews were placed as sub-codes within their respective high-level codes. At the end of multiple passes of analysis against the transcript, a set of root codes were created, with further sub-codes beneath each. Figure 2 shows a screenshot from the tool used during the coding phase of analysis, including the expert interview transcript and the passages that have been coded.

To create classification codes, the team analyzed each transcribed passage of the SME's response to the interview questions. Basic analysis was done by observing response words and context to determine what high-level SOC ontology hierarchy concepts were being discussed, then repeating this procedure to further identify and refine lower-level SOC concepts. In this research we first conducted an initial pass through the collected data in order to analyze single passages and determine whether an expert's response in that passage could be placed under a people-, process-, or technology-based classification. However, during this analysis we determined that these three classifications were insufficient in describing the entirety of the considerations being used by the experts. We determined that other high-level classifications could be created as a result of interview discussions including Motivation, External service(s), and SOC function.

We illustrate our method using the following passage from one expert interview. This passage read "the budget for the SOC determines a number of factors on the creation of the SOC," could not be considered any of PPT. We did not consider these to be classified into PPT. Therefore, a new top-level code—Motivation was needed.

Table 5. Summary Statistics of Experts' Experience and Qualifications

Measure	Category	Count	Percentage
Years' Experience	0–5	0	0.00%
	6–15	6	42.86%
	16+	8	57.14%
Number of Years Working in a SOC	0–3	1	7.14%
	4–10	6	42.86%
	11+	7	50.00%
Number of SOCs Worked in	1–2	10	71.43%
	3–4	3	21.43%
	5+	1	7.14%
Sub-Field	Incident Response	4	28.57%
	Forensics	2	14.29%
	Security Operations	2	14.29%
	Management	2	14.29%
	Threat Intelligence	2	14.29%
	Other	2	14.29%
Highest Degree	Bachelors	0	0.00%
	Masters	7	50.00%
	PhD or Equivalent	3	21.43%
	None or Unknown	4	28.57%

After the first pass through the data was completed, a second round of analysis was conducted, in order to drill deeper into the specific topics and ideas touched upon in each passage from the expert interviews, allowing an examination of individual concept. To illustrate, we use the example of cost considerations, which was cited by numerous experts. At the high level, these considerations would fall into the newly created "planning" consideration classification. However, there are specific considerations related to more specific concepts when examined at a more granular level. Such an examination reveals that the word "budget" implies monetary cost, therefore a sub-code could be established under planning named Cost. However, other cost considerations may include concern over threats to reputation (for example, form a data breach), and those associated costs. These considerations led to the creation of a sub-code under "motivation" called "threats." These new classification codes, and their implication for the body of SOC development knowledge, are discussed further in Section 3 below.

3 Results

Analysis of the data gathered from the SOC experts yielded a number of key discoveries and initial observations, as described in the following findings.

3.1 Experts Profile

The data we collected from participants revealed several key characteristics relevant to establishing their expertise in the field of security operations and SOC development (number of years in cybersecurity, number of SOCs worked in, degrees and certifications, etc.). Table 5 shows summary statistics of the experts' relevant experience and qualifications in the field.

Table 6. Summary Statistics of Organizations

Measure	Category	Count	Percentage
Organization Size	0–100	5	35.71%
	101–500	3	21.43%
	501+	6	42.86%
Organization Type/Sector	Public	7	50.00%
	Private	7	50.00%
IT Department Size	0–20	3	21.43%
	21–100	3	21.43%
	100+	4	28.57%
	Unknown	4	28.57%
SOC/Security Team Size	0–15	8	57.14%
	16–50	4	28.57%
	51+	2	14.29%

The experts interviewed for this research have extensive experience working in SOC environments, with exactly half having worked in SOCs for more than 10 years. More than half (57%) have been in the IT field generally more than 15 years. It is noteworthy that more than 70% of the experts have worked in only one or two SOCs total. It is not clear why this may be the case. It may speak to the relative newness of the SOC concept or the desire of experts to maintain continuity.

The data collected also revealed important information about the organizations in which the experts currently work, as shown in Table 6. In addition to speaking to the expertise and qualifications of the participants, this set of data also adds to the validity of the general applicability of the findings. The summary statistics of the organizations reveal that a wide range of institutions were included in the dataset. This variability applied to organizational size, sector, public/private affiliation, and other information.

The organization-level data reveal that a wide range of SOC are represented in this study. The most consistent trait is that most SOC teams are small in size, having fewer than 15 total staff members. This was the case regardless of the overall organization size or the size of the IT department of that organization.

3.2 Developing a SOC: Beyond PPT

While there was some level of variance in the experience of the expert participants in the research, and an even greater level of variance in the types of organizations for which they worked, there was remarkably little variance in the main themes and approaches that they share as being relevant to SOC development. This is notable, as it suggests that there is significant structure and consistency in the body of knowledge used for SOC development. Of course, there is significant variance within the main themes—no two experts have the same exact approach and solution for any given problem,

The primary common theme observed across the SOC experts was an adherence to PPT as the three pillars of SOC development. Occasionally different terminology was used (human resources vs. people) but the discussion was consistent. This is not surprising, as these areas are considered essential elements for developing a SOC or similar security team (see Abd Majid and Ariffin [2021]; Torres [2015]) and are also highlighted in commonly used industry best practice such as the NIST CSF. However, two additional high-level themes also emerged that are less cited in the literature. The first pertains to motivational issues—why is this SOC capability being developed by the organization it will serve, and the criteria that the organization uses to determine the final appearance of their SOC. Are issues of cost, compliance, or incident prevention important? How will these be addressed?

The issue of motivation was a critical concern for SOC developers regardless of other factors such as organizational size. For example, a repeated point concerning motivation was that organizations and the leaders tended not to talk about cybersecurity and the need for a SOC until an incident occurred. One interviewee who worked at a large organization in the government sector noted:

> ...Nobody cares about cybersecurity until the incident happens - that we've seen that - the company has no money for MFA and then they get hit by something and then a way the incident generates tons of funding.

This individual worked for a large government organization with significant financial resources, which may initially suggest that the ability to divert resources is the primary factor at work here. However, similar sentiments were expressed by others in much different organizational circumstances. An expert working at a startup level organization compared the SOC to other costs centers or outside repair services:

> ... So like you think about if by the end of a year we are going to be valuable because of this, or not. Would you want us to exist, or you don't even care, you know? Do you care about electrician? No, you don't care. You just need you come and say you thank you.

The above suggests that the action is the result of discrete events and the motivations that they provide, rather than the availability of resources. In each example it was implied that an incident would lead to more resources being put in to security. These observations also demonstrate that motivations remain consistent across organizational types. This is important in establishing the importance of motivation, as well as the general applicability of the developing knowledge framework shared by the experts.

While it is clear from the expert interviews that motivation is a critically important consideration for SOC development, it is equally important to add the caveat that the impact of motivational factors does vary substantially depending on several other factors. For example, cost is a motivation across organizations, but larger organizations will address this issue much differently than their smaller, more resource-constrained counterparts. This is an important reminder that developing a SOC capability is a complex, multi-faceted process.

The second high-level theme that emerged from the SOC experts, in addition to the well-established PPT, was planning issues. These were the items that would not necessarily be part of the SOC's daily functions once it became operational, but that nonetheless needed to be addressed for it to reach that point of initial operations. Often these planning concerns related to some future PPT consideration, but experts saw the planning of them as a separate function. Consider the following observations:

> One of the ways to organize this is to, before the SOC is even formed, put together a steering committee that is a small group of stakeholders who can represent the interests of the business and can articulate the major business objectives in forming and operating in the SOC.

and

> ...what I would use is, first of all, scoping what is expected from this organization, how [many] resources it can provide. So seeing...which monitoring services and incident response services and [how much the] organization can provide for staffing and for technology and for consultancy.

Those involved in developing SOCs place a very high value on the ability to establish a clear planning process from the very beginning—before any solutions for PPT are obtained. Without this critical planning the final makeup of the SOC will not be aligned with organizational expectations, and it will be difficult to receive resources. It would be more challenging to align the SOC with organizational expectations after processes are established and people or solutions are procured.

3.3 Five Pillars of SOC Development

PPT, motivation, and planning represent the main pillars of the SOC development process that our coding framework helped identify. They also serve as a useful starting point for developing a fuller framework of SOC development knowledge. As more of the interviews with SOC experts were examined in the coding process, we developed a coding framework with those five pillars at the roots of the framework. From there, branches emerged that demonstrate first the main considerations under each theme and then, further down the framework, the key individual issues.

Notably, the coding framework became saturated quite early in the coding process and has showed remarkable stability even as the interviews with new experts were added to the set of coded data. The first four high-level themes (PPT and motivation) were apparent in the first coded interview. The fifth (planning) was added in the coding of the second interview. Further down the framework, the "leaves" of the structure also eventually achieve stability, though much later. A total of 37 codes have been included in the framework, but only three of those were added after the first 10 interviews were coded.

The stability of the high-level themes (even as we continue to populate the lower levels) both affirms the importance of those themes and demonstrates the complex nature of SOC development. Even after several experts have been presented with a similar SOC development scenario, new additions are still made to the knowledge structure at lower levels.

Figure 3 depicts a partially developed knowledge tree structure showing how the people theme initially developed. The lower levels of the structure, which are the "leaves" of the knowledge tree (as opposed to the high-level "roots"), outline the important concepts and objects for that part of the structure. As the coding process continues to integrate expert knowledge, a full framework emerges to show which should address key questions, such as what challenges an organization would have in building a SOC capability, what tools, platforms, and infrastructure are needed to develop and operate a SOC capability, and what an organization's security priorities are when developing a SOC capability.

4 Discussion

The data gathered in this study demonstrate a clear knowledge structure for the process of developing a SOC capability in the service areas of incident response and network monitoring. The emerging coding framework demonstrates that these foundational elements link the overarching objective of SOC development to more specific concepts such as skills, policies, or tools. In this section we examine these findings, their implications, and potential impacts. We also outline limitations of the study and ideas for future work.

4.1 Implications

We found that we were able to describe a full, comprehensive process for developing a SOC capability by focusing on five main themes. Three of the these (PPT) are well discussed in the existing literature. Two others (motivation and planning) were not. From these five pillars, the remaining themes develop at a more granular level. We also found that these issues of motivation and planning are important primarily because organizations typically address the need for a SOC only after an incident has occurred or some outside requirement to develop one has been imposed.

This suggests that a more comprehensive approach to SOC development may be considered, as depicted in Figure 4. Existing approaches, which tend to focus solely on the PPT triad may be missing underlying themes, and such an approach may not result in the best outcome. There are implications for both the security professionals building and operating the SOC as well as for the leaders of the organization that the SOC will serve. For the security professionals, incorporating these pillars into the development process will require greater understanding of business goals and needs. For the organizational leader, it may be necessary to consider the reason a SOC is needed. As noted in Section 2.2 above, the data gathered in this study was developed form scenarios which

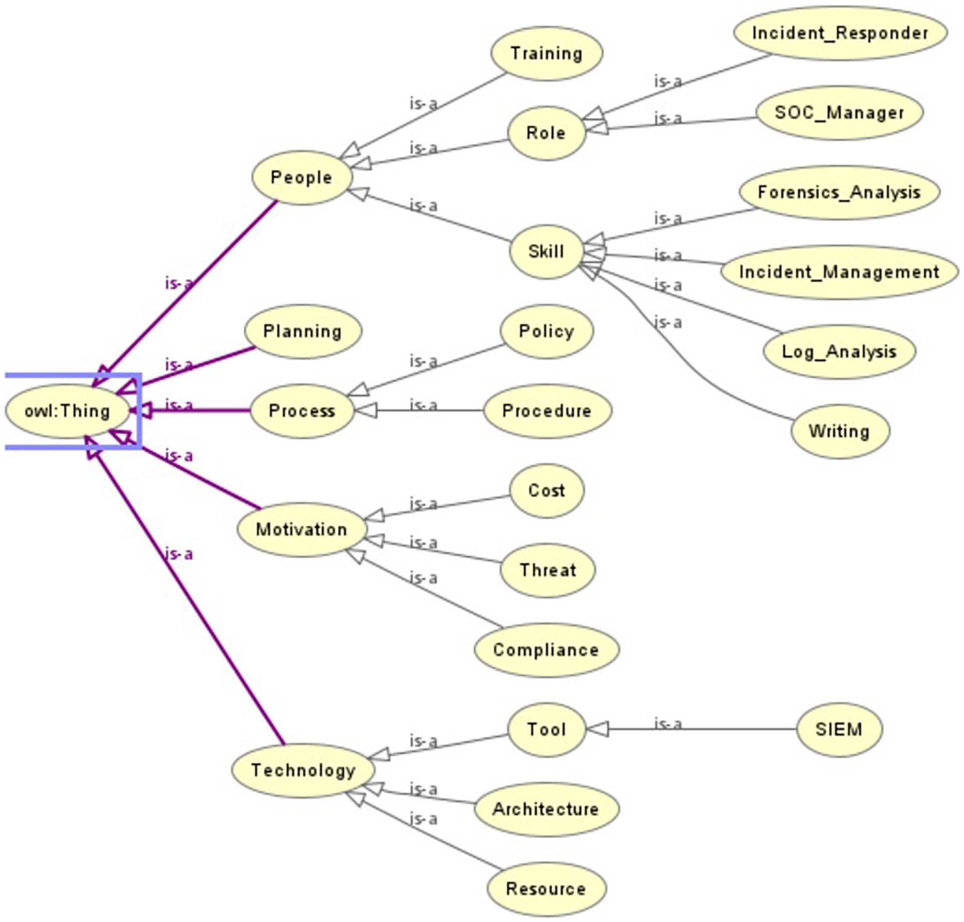


Fig. 3. Knowledge tree of people theme development.

were driven by either a regulatory requirement or the occurrence of an incident. Understanding how differing motivations will drive SOC development is a key consideration for leadership. There may also be additional motivations beyond these two which would further affect SOC development. This is an area for future research. Finally, we found it noteworthy that SOC experts tend to have limited perspectives—with most having only worked in one or two SOCs. Despite this, there is significant structure and consistency in the body of knowledge used for SOC development. Our coding frame reached saturation after less than a dozen interviews and remained stable after that.

The above findings have enabled us to begin building a structured body of knowledge on the subject of SOC development. We propose that this body of knowledge will represent an ontology, which we refer to as the OSCAR. The OSCAR ontology would represent the first attempt to collect and codify knowledge about SOC development in a usable format. A complete OSCAR ontology would be a valuable tool for both practitioners and researchers in the area of SOC development. For the practitioner, an ontology will serve as a resource, providing insight into the process of building a SOC and into the types of solutions for PPT that may be successful. For the researcher, it may serve as a starting point for understanding a variety of SOC issues and will also provide data and structured knowledge that can be used in future research.

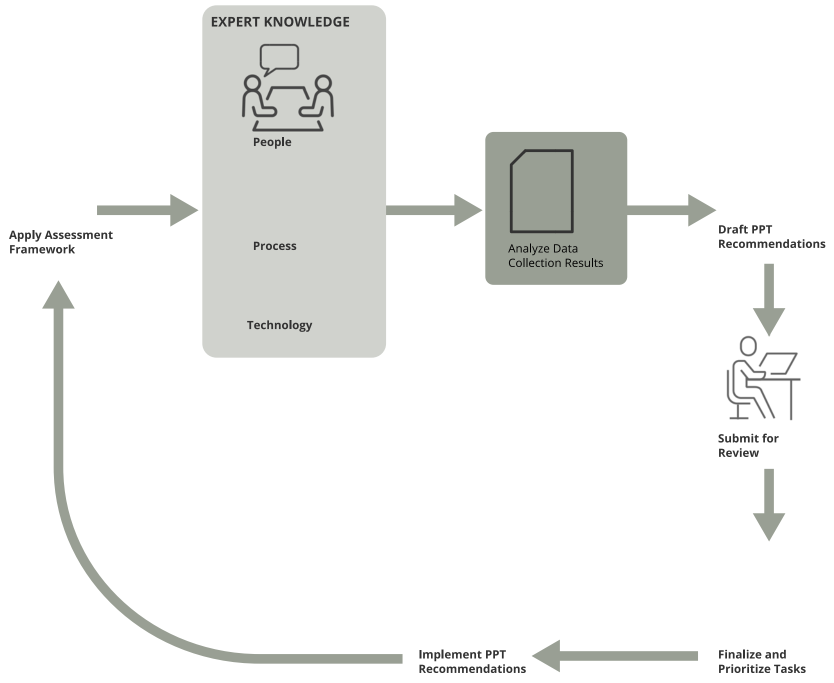

EXPERT KNOWLEDGE
People
Process
Technology
Apply Assessment Framework
Analyze Data Collection Results
Draft PPT Recommendations
Submit for Review
Finalize and Prioritize Tasks
Implement PPT Recommendations

Fig. 4. SOC development process.

4.1.1 A SOC Development Process. The research team involved in this study has significant field experience in developing and building cybersecurity teams (SOCs, CSIRTs, etc.) in a variety of settings. The lessons learned from those experiences are that no two SOCs are alike, and therefore the development of any SOC will be a unique iteration. Those field experiences are supported by the findings of this study. The data gathered here clearly indicates a set of pillars upon which any SOC must be built. It also demonstrates that there are common subjects or topics that must be addressed during SOC development—these are evident in the lower-level "leaves" of the knowledge structure, or in the code book that was developed during the study. However, despite this trove of data and experience, it is not possible to bring everything into a single, unified "theory of SOC development." There is no one-size-fits-all solution, and no single framework that will guide an organization through the process of building a SOC. SOC development is too unique and situationally specific.

Instead of a single framework or "SOC-in-a-box" solution then, what we are left with is a process. This process is repeatable but requires an expert hand to guide it. It begins with an assessment of an organization's current structure, continues with analysis of current and planned future capabilities for PPT, and may include a review of relevant documentation. From there, the involved SOC development experts will produce a series of proposed solutions in PPT, which can be vetted, reviewed, and updated as necessary. Finally, a plan or roadmap for creating the SOC is developed and implemented.

In this illustration of the SOC process, the need for expert knowledge is clear in areas such as describing the PPT required for SOC development. However, it is also required in other areas. For example, when applying an assessment framework at the beginning of the process, an organization must choose from a wide variety of options. However, the correct choice is context dependent—is the organization private sector or government? Must it comply with any regulations or standards? What is the goal of developing a SOC capability? In most cases, answering these questions requires the input of an expert who knows and understands SOCs well. It is clear from both research and practice that the role of the expert in this process is of paramount importance.

Such expertise, however, is difficult and time consuming for organizations to employ and may be prohibitive. The proposed OSCAR ontology replicates the knowledge and insights which may be offered by an expert, but at a lower barrier to use by an organization or practitioner. An ontology is significantly more robust than a framework. Where a framework such as NIST CSF or ISO 27000 series can offer a systemic approach to solving a particular problem, as an ontology OSCAR will help organizations and non-experts ask and answer the right questions and identify both the problems and the potential solutions which are context dependent and situationally unique. In this way, OSCAR may provide a valuable resource through the SOC development process.

4.1.2 Helping the Community Build Better SOCs. The sum result of this new insight and understanding of the pillars of SOC development is an improvement in the efficient building of more effective SOCs and SOC capabilities. Developing a SOC capability requires several things, but among the most important and an understanding of why the capability is needed, and an understanding of what specific solutions will address that need. Currently, understanding need is done by conducting some type of assessment or evaluation of the organization building the capability. The additional pillars presented here, along with the proposed ontology, will improve assessment and evaluation processes by helping to better understand motivations and planning processes, which have been under-explored in the past. And the re-defined pillars of SOC development will help in the identification of appropriate solutions by defining those needs with greater granularity and specificity, enabling more refined results.

4.2 Limitations

This data gathering study has provided a new dataset for understanding SOC development and has further developed that data into a formal knowledge structure. We propose an ontology as the next step in the formalization of that knowledge, but do not present a full ontology yet here. Beyond this, there are some specific limitations which constrain the general applicability of this study. First, the study design was intentionally limited to only two areas of SOC operations—incident response and network monitoring. Therefore, we cannot say conclusively that our finding will apply to other SOC functions or to more general SOC operations. Moreover, the generalization of our findings may be limited by the sample of experts that we used to gather data. Despite the extensive measures taken to include participants from a wide range of backgrounds, our small sample size means that some perspectives may not be included. Second, there was a notable lack of sub-codes beneath one of the five pillars identified by the research, namely planning. It is unclear why this is the case. During the coding process, "planning" originally appeared to be a part of the "process" pillar before it became evident that it was its own distinct pillar. It may be the case that planning is simply a distinct discipline from information security or security operations. However, our study did not ask participants about background or expertise in planning and they therefore may have less insight on the subject. This is an area that requires further investigation.

4.3 Future Work

In this study, we examine the process for developing a SOC capability and create a coding framework based on the knowledge developed from that examination. The primary avenue of future research will be to continue developing the knowledge base into the proposed OSCAR ontology and then to integrate that knowledge base into a more refined decision- making tool, such as an expert system. The ontology that we have proposed here would represent a significant advance in the field and would be a valuable tool for practitioners. An expert system for SOC development would represent an additional advance.

Two new areas of knowledge are identified in this study—the SOC development pillars of planning and motivation, which go with the three pillars identified in existing research—PPT. In the future, these new pillars can be researched and understood much more extensively now that they have been identified. More applications may be developed for understanding and implementing them in practice.

Finally, to make the findings of this study more generally applicable, future work may expand to include the study of additional SOC services beyond just the two addressed by this study: Network Monitoring and Incident Response. Eventually, the resulting knowledge base could be large and robust enough to be generally applicable to any SOC capability.

5 Conclusions

The growth and proliferation of SOCs and similar cybersecurity bodies within a variety of organizations is set to continue amid a seemingly exponential increase in cybersecurity threats. On one hand, this has led to notable growth and development of SOC-specific expertise even within the cybersecurity community. On the other hand, the needs of organizations to deploy SOCs have continued to outpace the growth of that expertise. This suggests an approach for bridging the gap between needed and available SOC development expertise. It does so by demonstrating an approach for gathering and codifying SOC development knowledge, and then proposing a formal ontology to describe the knowledge in a format accessible to a wider range of individuals.

This proposed ontology would provide a formally defined knowledge domain for SOC development encompassing expected categories of knowledge—PPT—as well as newly defined categories of knowledge, such as motivation and organizational planning tasks. In this study, we were able to explore these categories of knowledge both at a high level and in greater depth by using a qualitative data gathering approach centered on interviewing the actual experts in a way designed to capture their individual knowledge and insights. We developed a systematic approach to create scenarios which would encourage openness and frankness as experts explored how they develop SOC capabilities and solve problems which arise during the process. Across 14 interviews, we built a database of more than 500 minutes of interviews totaling more than 30,000 words of expert insight on SOC development.

We found that there were clear themes within SOC development that were true regardless of some factors such as organization size, budget, or regulatory requirements. These common themes included a primary focus on the role of people in establishing an effective SOC, including a prioritization of soft skills over hard technical skills; an acknowledgement that budget and high-level organizational prioritization were critical to success; and a view that technology was not a panacea to solve all problems. On the other hand, we also found that some things varied depending on context and situation. The ability to scale and the services offered by SOCs in particular depending on size of organization, type of organization, and sector (public vs. private).

The sum total of these insights is that it is not possible to establish a simple, singular framework for establishing a SOC. Instead, a more flexible process can be defined and then used by interested parties to develop a SOC capability. In order to define and make usable such a process, a foundational body or structured knowledge is required. This research proposes the OSCAR ontology as a potential solution for this body or structured knowledge.

This study is limited in that we are not yet able to present a fully developed ontology or to integrate it into any type of more complete tool. Further, our research has focused narrowly on only two SOC capabilities—network monitoring and incident response. To provide a truly useful solution, future research can focus on completing the ontology in these areas.

References

Maziana Abd Majid and Khairul Akram Zainol Ariffin 2021. Model for successful development and implementation of Cyber Security Operations Centre (SOC). *PLoS One* 16, 11 (2021), e0260157. DOI: https://doi.org/10.1371/journal.pone.0260157

BrigitteS Cypress. 2015. Qualitative research. *Dimensions of Critical Care Nursing* 34, 6 (2015), 356–361. DOI: https://doi.org/10.1097/DCC.0000000000000150

JulieG Donalek.2005. The interview in qualitative research. *Urologic Nursing* 25, 2 (2005), 124–125. DOI: https://doi.org/15900982

FIRST—Forum of Incident Response and Security Teams. 2023. CSIRT Services Framework Version 2.1. Retrieved November 27, 2023 from https://www.first.org/standards/frameworks/csirts/csirt_services_framework_v2.1

FIRST—Forum of Incident Response and Security Teams. 2024. Team Types within the Context of Services Frameworks. Retrieved November 11, 2024 from https://www.first.org/standards/frameworks/csirts/team-type#:~:text=All%20four%20team%20types%20(i.e.,incident%20management%20or%20security%20capability

Ralph Grishman, Silja Huttunen, and Roman Yangarber. 2002. Information extraction for enhanced access to disease outbreak reports. *Journal of Biomedical Informatics, Sublanguage—Zellig Harris Memorial* 35, 4 (August 2002), 236–246. DOI: https://doi.org/10.1016/S1532-0464(03)00013-3

H. Holbrook. 1990. A scenario-based methodology for conducting requirements elicitation. *ACM SIGSOFT Software Engineering Notes* 15, 1 (January 1990), 95–104. DOI: https://doi.org/10.1145/382294.382725

ShahidN Khan. 2014. Qualitative research method: Grounded theory. *International Journal of Business and Management* 9, 11 (2014). DOI: https://doi.org/10.5539/ijbm.v9n11p224

Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. 2019. Matched and mismatched SOCs: A qualitative study on security operations center issues. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1955–1970. DOI: https://doi.org/10.1145/3319535.3354239

Steve Mansfield-Devine. 2016. Creating security operations centres that work. *Network Security* 2016, 5 (May 2016), 15–18. DOI: https://doi.org/10.1016/S1353-4858(16)30049-6

Arif Ali Mughal. 2022. Building and securing the modern Security Operations Center (SOC). *International Journal of Business Intelligence and Big Data Analytics* 5, 1 (2022), 1–15.

Helen Noble and Gary Mitchell. 2016. What is grounded theory? *Evidence-Based Nursing* 19, 2 (April 2016), 34–35. DOI: https://doi.org/10.1136/eb-2016-102306

Cyril Onwubiko and Karim Ouazzane. 2022. SOTER: A playbook for cybersecurity incident management. *IEEE Transactions on Engineering Management* 69, 6 (December 2022), 3771–3791. DOI: https://doi.org/10.1109/TEM.2020.2979832

Rob Van Os. 2023 SOC-CMM—Measuring Capability Maturity in Security Operations Centers. Retrieved March 15, 2023 from https://www.soc-cmm.com/

Rodney Petersen, Danielle Santos, Matthew C. Smith, Karen A. Wetzel, and Greg Witte. 2020. *Workforce Framework for Cybersecurity (NICE Framework)*. National Institute of Standards and Technology. DOI: https://doi.org/10.6028/NIST.SP.800-181r1

Jennifer Rowley. 2012. Conducting research interviews. *Management Research Review* 35, 3/4 (March 23, 2012), 260–271. DOI: https://doi.org/10.1108/01409171211210154

Daniel Schlette, Manfred Vielberth, and Günther Pernul. 2021. CTI-SOC2M2—The quest for mature, intelligence-driven security operations and incident response capabilities. *Computers & Security* 111 (December 2021), 102482. DOI: https://doi.org/10.1016/j.cose.2021.102482

James Shanteau. 2015. Why task domains (still) matter for understanding expertise. *Journal of Applied Research in Memory and Cognition, Modeling and Aiding Intuition in Organizational Decision Making* 4, 3 (September 1, 2015), 169–175. DOI: https://doi.org/10.1016/j.jarmac.2015.07.003

Don Stikvoort. 2015. *SIM3: Security Incident Management Maturity Model*. Open CSIRT Foundation.

Stephanie Susnjara and Ian Smalley. 2024. What Is the IT Infrastructure Library (ITIL)? Retrieved October 9, 2024 from https://www.ibm.com/topics/it-infrastructure-library

Steven Tenny, Janelle Brannan, and Grace Brannan. 2022. Qualitative Study. NIH. National Library of Medicine. Retrieved September 2022 from https://www.ncbi.nlm.nih.gov/books/NBK470395/

Chooi Shi Teoh, Ahmad Kamil Mahmood, and Suhazimah Dzazali. 2018. Cyber security challenges in organisations: A case study in Malaysia. In *Proceedings of the 2018 4th International Conference on Computer and Information Sciences (ICCOINS)*. IEEE, 1–6. DOI: https://doi.org/10.1109/ICCOINS.2018.8510569.

Alissa Torres. 2015. *Building a World Class Security Operations Center: A Roadmap*. White Paper. SANS Institute.

Hersh Trivedi and Katherine H. Chan. 2023. Quality over quantity: How qualitative research informs and improves quantitative research. *Journal of Pediatric Urology* 19, 5 (October 2023), 655–656. DOI: https://doi.org/10.1016/j.jpurol.2023.05.007.

Manfred Vielberth, Fabian Bohm, Ines Fichtinger, and Gunther Pernul. 2020. Security operations center: A systematic study and open challenges. *IEEE Access* 8 (2020), 227756–227779. DOI: https://doi.org/10.1109/ACCESS.2020.3045514

Appendices

A Sample Interview Questions and Scenarios

The below interview and scenario descriptions were used to gather data from study participants:

This interview will seek to understand SME perspectives and approaches to developing SOCs.

Preliminary questions. Please do not reveal any private or personally identifiable information about yourself OR others in your responses.

- *What is your specific field of expertise?*
  - *Number of years of experience?*
  - *What Education/Certifications do you have?*
- *What sector do you work in? Please select one or more of:*
  - *Government*
  - *Private Sector*
  - *Critical Infrastructure*
  - *Other*
- *What is the size of your organization?*
  - *How many people?*
  - *Number of IT assets?*
  - *Annual budget for IT operations and/or security?*
- *Are you currently working in a SOC or other incident response function/team?*
  - *How long has your team been in operation?*
  - *How many staff are assigned to your team?*
  - *How many total personnel are in your organization?*
  - *How many SOCs have you worked in?*

Based on responses to the above questions, you will be asked to consider one or two of the following scenarios, each of which has to do with the creation of a SOC. In each case, an on-premises SOC must provide incident response and network monitoring services. Other services are optional.

SOC Scenarios

- *Scenario 1—Small government experiences APT attack*
- *Scenario 2—Medium government experiences APT attack*
- *Scenario 3—Large government experiences APT attack*
- *Scenario 4—Small CI operator experiences APT attack*
- *Scenario 5—Medium CI operator experiences APT attack*
- *Scenario 6—Large CI operator experiences APT attack*
- *Scenario 7—Small Private Company experiences APT attack*
- *Scenario 8—Medium Private Company experiences APT attack*
- *Scenario 9—Large Private Company experiences APT attack*
- *Scenario 10—Small government has regulatory requirement to implement SOC*
- *Scenario 11—Medium government has regulatory requirement to implement SOC*
- *Scenario 12—Large government has regulatory requirement to implement SOC*
- *Scenario 13—Small CI operator has regulatory requirement to implement SOC*
- *Scenario 14—Medium CI operator has regulatory requirement to implement SOC*
- *Scenario 15—Large CI operator has regulatory requirement to implement SOC*
- *Scenario 16—Small Private Company has regulatory requirement to implement SOC*

- *Scenario 17—Medium Private Company has regulatory requirement to implement SOC*
- *Scenario 18—Large Private Company has regulatory requirement to implement SOC*

After the scenario has been presented, the research team will ask a series of questions designed to help you better understand how you might go about developing a SOC to meet the needs of the organization in the given scenario. Your answers will help to identify the PPT required to develop a SOC capability, and to understand the decision-making processes and approaches to prioritization used by SOC developers and operators.

The questions asked during the interview will depend on your responses throughout, but the general topics covered will include those below:

(1) *What motivates an organization to build a SOC capability?*
   - *Why do you need a SOC?*
   - *When building a SOC, how does an organization determine and validate needs, goals, objectives, and so on?*
   - *How do you determine whether tools are effective and goals and objectives are being met?*
   - *How frequently do you re-evaluate?*

(2) *What challenges would an organization have in building a SOC capability?*
   - *Discuss areas such as funding, staffing, workforce (development and retention), skills, tool acquisition, and so on.*
   - *How do you prioritize/weigh which tools are needed?*

(3) *What tools, platforms, and infrastructure are needed to develop and operate a SOC capability?*
   - *How do you identify which tools are needed?*
   - *Are third-party capabilities needed?*

(4) *What is needed for organizations to develop or implement SOC PPT?*
   - *What policies, procedures, and so on are required?*
   - *What about governing articles such as Charter, Playbooks, and so on?*
   - *How do you determine what people are needed, and what skills and knowledge they require?*

(5) *What are an organization's security priorities when developing a SOC capability?*
   - *What do you do once you have a SOC capability?*
   - *Who develops the priorities, and where do they come from?*
   - *How does this affect organizational issues such as data segmentation?*
   - *How often are these priorities reviewed?*

B Coding Frame

The full list of codes, including the number of interviews in which each code was referenced, and the total number of references for each code across all interviews, is in the following Table B1.

Table B1. Full List of Referencing Codes

Name	Interviews Referencing Code	Total References
External Service	8	30
Malicious Activity	4	7
Motivation	5	40
Compliance	10	35
Cost	9	62
Incident Prevention	5	13
Leadership	5	15
Security Process Improvement	9	39
Threats	10	28
Organization	7	39
Capacity	5	20
Number of Assets	4	4
Number of Users	6	11
Size of IT Department	2	3
Size of Security Team	9	19
Size of Organization	6	7
People	8	57
Experience Level	11	53
Nationality	1	2
Role	9	55
Sector Expertise	7	13
Skill	6	14
Soft Skill	10	49
Technical Skill	11	55
Time in Role	6	10
Training	7	20
Certification	9	13
Degree	9	15
Practice	4	7
Workload	5	10
Planning	10	128
Process	6	129
Asset Management	4	7
Communications	5	20
Lessons Learned	5	18
Metrics	10	33
Policies	9	72
Procedure	11	57
Testing and Validation	5	16
Scenario	5	8
SOC Function	12	53
Technology	6	59
Architecture	8	23
Resource	11	63
Tools	11	120

Received 30 December 2023; revised 11 February 2025; accepted 27 February 2025

ACM Digital Threats: Research & Practice
https://dtrap.acm.org/

Guide to Manuscript Submission
Submission to the *ACM Digital Threats: Research & Practice* is done electronically through https://mc.manuscriptcentral.com/dtrap. Once you are at that site, you can create an account and password with which you can enter the ACM Manuscript Central manuscript review tracking system. Proceed to the Author Center to submit your manuscript and your accompanying files.

You will be asked to create an abstract that will be used throughout the system as a synopsis of your paper. You will also be asked to classify your submission using the ACM Computing Classification System through a link provided at the Author Center. In addition to clarifying the area where your paper belongs, classification often helps in quickly identifying suitable reviewers for your paper.

The ACM Production Department prefers that your manuscript be prepared in either LaTeX or MS Word format. Style files for manuscript preparation can be obtained at the following location: https://www.acm.org/publications/authors/submissions. For editorial review, the manuscript should be submitted as a PDF or Postscript file. Accompanying material can be in any number of text or image formats, as well as software/documentation bundles in zip or tar-gzipped formats.

Questions regarding editorial review process should be directed to the Editor-in-Chief. Questions regarding the post-acceptance production process should be addressed to the Associate Director of Publications, Sara Kate Heukerott, at heukerott@hq.acm.org.

Subscription and Membership Information.
All papers published in ACM's Digital Threats: Research & Practice (DTRAP) will be published as Gold Open Access (OA) and will be free to read and share via the ACM Digital Library. For an initial two-year period DTRAP authors will not be asked to pay an Article Processing Charge (APC) to underwrite the cost of OA publication, as these costs will be underwritten by ACM. For additional information about ACM's various Open Access publication initiatives, visit https://www.acm.org/publications/openaccess.

For information, contact:

ACM Member Services Dept.
1601 Broadway, 10th Floor
New York, NY 10019-7434
Phone: +1-212-626-0500
Fax: +1-212-944-1318
Email: acmhelp@acm.org
Catalog: https://www.acm.org/publications/alacarte

dtrap.acm.org

Open Access

acm Association for Computing Machinery