electronics

MDPI

Article

Clustering Network Traffic Using Semi-Supervised Learning

Antonina Krajewska [1] and Ewa Niewiadomska-Szynkiewicz [2,*]

[1] Research and Academic Computer Network (NASK), Kolska 12, 01-045 Warsaw, Poland; antonina.krajewska@nask.pl
[2] Institute of Control and Computation Engineering, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
* Correspondence: ewa.szynkiewicz@pw.edu.pl; Tel.: +48-884-260-036

Abstract: Clustering algorithms play a crucial role in early warning cybersecurity systems. They allow for the detection of new attack patterns and anomalies and enhance system performance. This paper discusses the problem of clustering data collected by a distributed system of network honeypots. In the proposed approach, when a network flow matches an attack signature, an appropriate label is assigned to it. This enables the use of semi-supervised learning algorithms and improves the quality of clustering results. The article compares the results of learning algorithms conducted with and without partial supervision, particularly non-negative matrix factorization and semi-supervised non-negative matrix factorization. Our results confirm the positive impact of labeling a portion of flows on the quality of clustering.

Keywords: cybersecurity; honeypot; semi-supervised learning; clustering

check for updates

Citation: Krajewska, A.; Niewiadomska-Szynkiewicz, E. Clustering Network Traffic Using Semi-Supervised Learning. *Electronics* **2024**, *13*, 2769. https://doi.org/10.3390/electronics13142769

Academic Editor: Andreas Mauthe

Received: 28 May 2024
Revised: 3 July 2024
Accepted: 12 July 2024
Published: 14 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The pervasive growth of information and communication technologies (ICTs) has rendered them increasingly susceptible to cyberattacks. In response, engineers and security specialists have developed new methods and tools to protect telecommunication infrastructure. Cybersecurity solutions, ranging from advanced encryption algorithms to artificial intelligence (AI)-based threat detection systems, are indispensable for handling increasingly complex threats [1–3].

Distributed honeypot systems serve as potent aids in enhancing situational awareness. By simulating the behavior of IT systems, honeypots entice and monitor potential cybercriminal activities. These distributed systems can analyze vast amounts of traffic data from diverse sources in real time, bolstered by advanced machine learning (ML) algorithms. Thus, these systems enable the identification of attack patterns and the adaptation to evolving adversary tactics, consequently fortifying ICT systems and allowing security specialists to respond swiftly to novel threats [4,5].

Clustering algorithms play a central role in the efficacy of such systems [6]. They are pivotal in detecting new attack patterns and anomalies while enhancing overall system performance. The rapid advancement of AI techniques, particularly semi-supervised learning, has significantly augmented the effectiveness of these methods. Clustering is particularly useful in analyzing malicious network traffic from distributed honeypot systems, as it enables the generation of signatures for previously unknown attacks [7,8]. However, some connections with honeypots are examples of known attacks that match signatures integrated with Intrusion Detection Systems (IDSs).

Applying matrix factorization with constraints offers a unified framework for expressing diverse clustering algorithms [9]. Among these techniques, non-negative matrix factorization (NMF) is a prominent method that leverages constraints to extract meaningful patterns from data across various domains.

In recent years, semi-supervised clustering has received considerable attention. These algorithms, which harness partial labeling to guide the grouping process, can enhance

clustering malicious traffic analysis. In this paper, we propose utilizing an algorithm with partial supervision. Our motivation stems from the observation that certain connections with honeypots represent recognizable traffic patterns, such as established attacks or benign activities. By tracing these patterns, we can enhance the effectiveness of IDS signatures through the application of semi-supervised clustering techniques [10].

This paper presents a method for clustering malicious network traffic in a semi-supervised manner. Our motivation comes from observing that certain connections made with honeypots match well-documented attacks. Consequently, flows associated with a specific attack should be grouped within a cluster. On the other hand, flows resembling the signature of another attack should not be erroneously included in the same cluster. Our approach also aids clustering by using signatures that define benign traffic.

The main contributions of this paper are summarized as follows:

1. We introduce a novel approach for clustering malicious network traffic using partially labeled traffic, referred to as SCOUT (Semi-supervised Clustering Of Untrustworthy Traffic);
2. We utilize a non-negative matrix factorization-based algorithm for clustering;
3. We propose the architecture of the Network Traffic Analysis Cluster integrated with SCOUT;
4. We perform numerous experiments to validate our approach.

The rest of the paper is organized as follows. Section 2 reviews the literature. Section 3 focuses on the significance of network traffic analysis, the importance of accurate data representation, and the role of clustering network traffic in cybersecurity systems. We also motivate employing semi-supervised clustering in distributed honeypot systems. Section 4 provides the problem formulation and introduces the SCOUT method. Section 5 compares unsupervised and semi-supervised clustering methodologies. We propose the architecture of the Network Traffic Analysis Cluster in Section 6 and discuss the key challenges and limitations of the proposed approach in Section 7. We outline the experimental procedures in Section 8. The experimental results and subsequent discussions are presented in Section 9. Finally, Section 10 concludes our study and provides future research directions.

In this paper, we use the following notation. Vectors are denoted by lowercase boldface letters such as $\mathbf{x}$, and matrices are denoted by uppercase boldface letters such as $\mathbf{X}$. The set of $n \times m$ real-valued matrices with non-negative elements is denoted by $\mathbb{R}^{n \times m}_{\geq 0}$. In optimization problem formulations, for simplicity, we omit matrix dimensions and refer to a matrix with non-negative elements as $\mathbf{X} \geq 0$. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, its Frobenius norm is defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}^2}$, where x_{ij} is its (i, j)-th element. For two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n \times m}$, their Hadamard product is denoted by $\mathbf{X} \odot \mathbf{Y}$ and is given by $(\mathbf{X} \odot \mathbf{Y})_{ij} = x_{ij} y_{ij}$.

2. Related Works

In recent years, there has been a growing interest in incorporating machine learning (ML) methods into cybersecurity systems [1,4,5,8,11–17]. These models leverage various data sources, including logs from honeypot software and network traffic. Numerous ML models have been developed to detect patterns in data collected from SSH-emulating honeypots [4,5,13–15]. Satria et al. proposed a model for generating signatures of brute-force attacks based on connection frequency [14]. Many studies focus on session analysis, particularly on examining SSH commands [4,5,13,15]. They employ AI algorithms like Support Vector Machines (SVMs) [4] and GPT-2 [5] for log analysis. Notably, the CYBEX-P system aims to predict future commands [15].

Research also emphasizes ML algorithms for analyzing network traffic data [1,3,16,17]. Krzysztoń et al. proposed a one-class classifier for anomaly detection [16]. Sharafaldin et al. benchmarked widely recognized classification algorithms for attack detection within Intrusion Detection Systems (IDSs) [1].

In IDSs, clustering techniques are crucial for automated signature generation to identify attack patterns [12]. These methods typically employ content-based approaches, such as detecting substrings within packet payloads. Werner utilized hierarchical clustering

for distance-based signature generation [8], while Perdisci et al. introduced a behavioral malware clustering system that was tailored for detecting malicious HTTP traffic [18].

Tropp demonstrated that many clustering algorithms can be expressed as matrix factorization problems with constraints [19]. Non-negative matrix factorization (NMF) stands out as a prominent clustering method [20,21], with Udell extending NMF to handle arbitrary datasets [9]. Semi-supervised learning algorithms have gained popularity [10,22–25], operating on partially labeled data. Semi-supervised Non-negative matrix factorization (SSNMF), introduced by Lee et al. [24] and extended by Haddock et al. [23], has been examined for clustering malicious network traffic [26].

3. Network Traffic Representation and Clustering

While honeypots offer valuable insights into network events and attacker behavior, the data they capture are limited and prone to errors. Logged information from honeypot software only partially represents the complete network traffic record. It is subject to variations in format and content, depending on the honeypot version and potential software errors. Analyzing raw network traffic records is crucial to understanding and fully enhancing situational awareness. Technologies like Arkime 5.0 (https://arkime.com/ (accessed on 27 May 2024)) , Snort 3.0 (https://www.snort.org (accessed on 27 May 2024)), Ntopng 6.1 (https://www.ntop.org (accessed on 27 May 2024)), or CicFlowMeter 4.0 (https://github.com/ahlashkari/CICFlowMeter (accessed on 27 May 2024)) offer efficient ways to process vast amounts of network traffic.

Additionally, the distinct characteristics of network traffic from honeypots make it a valuable dataset for AI algorithms. Honeypots support specific protocols such as HTTP, HTTPS, FTP, or SSH. As a result, the data have a well-defined, predictable characteristic. Since this traffic is assumed to be malicious, the dataset can be utilized, in particular, by unsupervised learning algorithms.

3.1. Network Traffic Representation

The quality of analysis results relies on the chosen representation of network traffic [27]. Two predominant approaches are commonly discussed in the existing literature: packet-based and flow-based. In the packet-based approach, individual network packets' headers and payloads are analysed, with data captured in pcap format. The header metadata vary depending on the network and transport protocols, such as TCP, UDP, ICMP, and IP [28]. As a reliable transport protocol, TCP encompasses metadata like sequence number, acknowledgement number, TCP flags, and checksum values. In contrast, UDP, a connection-less transport protocol, features a smaller header than TCP, comprising only four fields: source port, destination port, length, and checksum.

Flow-based data, on the other hand, represent aggregated packet data [29]. The prevalent aggregation criterion typically involves a set of five features: source IP address, source port, destination IP address, destination port, and protocol (Figure 1). Both unidirectional and bidirectional flows are distinguished [30]. In addition to the mentioned attributes, flow-based data include statistics such as communication duration, the number of bytes sent per second, time intervals between sent messages, and the size and number of transmitted packets [1]. Common flow-based formats include NetFlow [31], IPFIX [32], sFlow [33], and OpenFlow [34]. The more sophisticated yet flexible definition of network flow is used by FLDX, a nationwide cybersecurity system managed by the NASK National Research Institute [29]. Furthermore, other data formats exist, such as flow-based datasets enriched with additional information from packet-based data or host-based log files [30].

The advantages of representing network traffic in the form of flows include lower data processing and storage costs and less susceptibility to outliers. The system processes and stores less-sensitive data since the data do not include payloads. In the case of encrypted communication, there is no need to decrypt packet payloads. Such representation of network traffic is naturally mapped in the vector space. As a result, applying ML algorithms does not require complicated data preprocessing. Finally, as flow-based data are already

aggregated, they provide valuable contextual information for ML algorithms, enhancing their ability to derive meaningful insights.

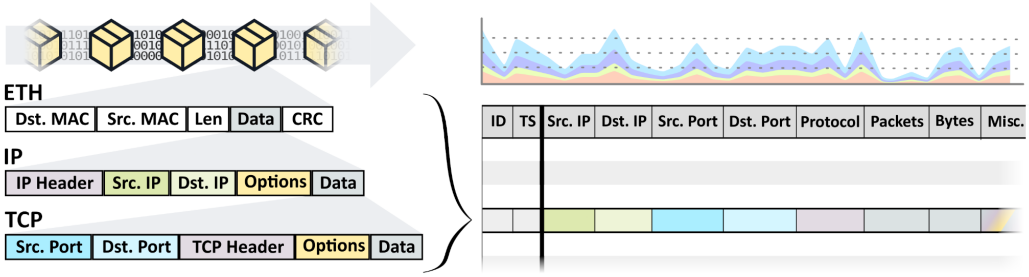


Figure 1. Packet-based and flow-based network traffic representations. On the **left**, individual Ethernet, IP, and TCP headers are detailed. On the **right**, packets are aggregated into flows using source IP (green), destination IP (light green), source port (blue), destination port (light blue), and protocol (heather), enabling efficient time series analysis. Each field is marked with different colors to improve readability.

3.2. The Role of Clustering Network Traffic

Clustering, as an unsupervised learning method, is fundamental in cybersecurity systems, which typically utilize one or both of the following techniques:

- Anomaly analysis, identifying threats by detecting deviations from a known normal profile of network activity in the monitored network [2];
- Using pre-generated rules (patterns, signatures) to detect malicious content in network flows [12].

In anomaly detection, clustering is a foundational step. It enables the identification of unusual patterns that are indicative of potential threats by grouping typical profiles or unusual behaviors. On the other hand, signatures are formal descriptions of the distinctive characteristics of a specific campaign, encapsulating the patterns of network attacks. For instance, the Nebula algorithm, proposed by Werner et al. [8], initially groups packets of malicious network traffic. Subsequently, it generates a set of signatures describing each cluster. These signatures can be expressed in a format supported by IDSs like Snort. Among metadata, they consist of sequences of payload packet fragments with defined allowable start and end positions in the payload (Figure 2).

```
alert tcp any any -> $HOME_NET 8000 (msg: "Authomaticaly Generated Rule 2000005 rev. 1";
content: "GET / HTTP/1.0|0d 0a|User-Agent\: DFXPDFXPAAA|eb 03|Y|eb 05 e8 f8 ff ff ff|II"; offset: 0; depth: 51;
content: "|41 42 43 44|"; offset: 51; depth: 4;
content: "|4D 45 54 48 4F 44 20|"; offset: 55; depth: 7;
content: "|58 50 38|"; distance: 18; within: 525;
content: "|0d 0a|Authorization\: Basic UVVGQlFVRkJRVUZCUVVGQlFVRkJRVUZCUVVGQlFVR
VUZCUVVGQlFVRkJRVUZCUVUZCUVVGQlFVRkJRVUZCUVVGQlFVRkJRVUZCUVVGQlFVRkJRVUZCUVVGQ
lFVRkJRVUZCUVVGQm5TOEFFSkNRa0pBejBtYUJ5djhQUWxKcUFsak5MandGV25UdnVFUkdXRkNMK3E5MTZxOTE1Ly9uUWtKQ1FrSkNRa0pDUWtKQ1FrSkNR
a0pDUWtKQ1FrSkNRa0pBejBtYUJ5djhQUWxKcUFsak5MandGV25UdnVFUkdXRkNMK3E5MTZxOTE1Ly9uUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWt
KQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1Fr
NRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pD
UWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ
1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkNRa0pDUWtKQ1FrSkN
Ra0pDUWtKQ1FrSkNRa0pDUWtKQ1FrST0=|0d 0a 0d 0a|"; distance: 58; within: 1845;
sid: 2000005; rev: 1;)
```

Figure 2. Snort rule.

The most common use case for such generated signatures is their automatic deployment in IDSs or Intrusion Prevention Systems (IPSs) to halt outbreaks of previously unknown threats. Such usage requires signatures of very high quality, which depend on the clustering algorithms used. However, clustering flows, or network packets, also improve the quality of signatures for known attacks and facilitate the system operator's classification of network events. Network traffic that the system receives does not necessarily have to be an attack. Some connections to honeypots result from the misconfigurations of network devices. Effective clustering algorithms facilitate determining whether a given network event is an attack requiring further analysis.

4. Problem Formulation and SCOUT Method Overview

In the domain of network security, various solutions utilize machine learning techniques for flow-based Intrusion Detection Systems (IDSs), as well as for network monitoring and analysis. While many of these solutions operate without relying on predefined signatures, some incorporate signature-based engines to enhance their threat detection capabilities. For example, Cisco Secure Network Analytics (https://www.cisco.com (accessed on 27 May 2024), version 7.5.0). enhances threat visibility using behavioral modeling, multilayered machine learning, and global threat intelligence without relying on signatures. In contrast, the Flowmon Anomaly Detection System (https://www.flowmon.com (accessed on 27 May 2024), version 12.3) employs artificial intelligence and machine learning to identify anomalies in network traffic and supports integration with Suricata 7.0.6 (https://suricata.io/ (accessed on 27 May 2024)) IDS alerts, highlighting the advantages of combining signature-based and anomaly-based detection methods.

In this context of advanced network security solutions, we introduce SCOUT (Semi-supervised Clustering Of Untrustworthy Traffic), a novel method for clustering malicious traffic represented by partially labeled flows. The labels indicate association with a particular attack and can be assigned either by an Intrusion Detection System (IDS) or manually by a cybersecurity expert. The traffic data can be collected through methods such as a distributed honeypot system. SCOUT leverages labels to guide the clustering of suspicious traffic represented as flows. Flows associated with a specific attack are grouped within a cluster, while flows that resemble the signature of another attack are not erroneously included in the same cluster (Figure 3).

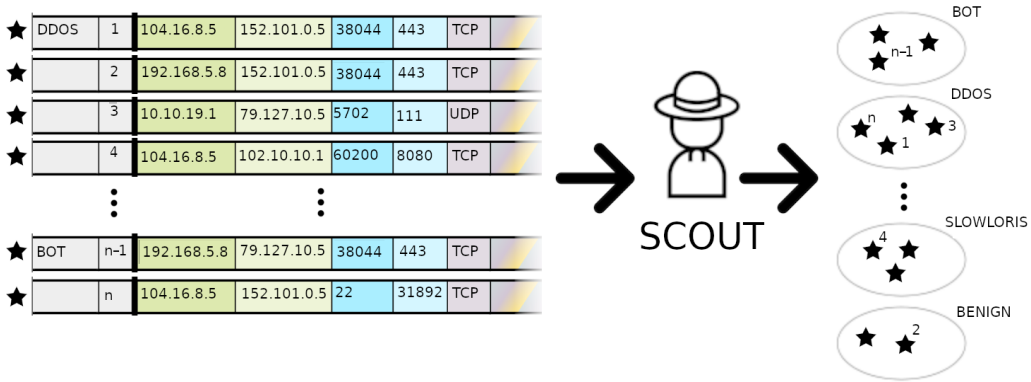


Figure 3. Overview of the SCOUT method for clustering partially labelled flows, represented by a star symbol.

The SCOUT method offers several advantages and meets critical objectives:

- Support for cybersecurity analysis (offline mode): SCOUT is designed to handle a substantial volume of malicious flows daily. It operates effectively with partially labeled data, ensuring robustness even when only a small portion of the data are labeled. This feature is crucial for practical deployment, where exhaustive labeling by security experts is impractical. In a scenario where an organization collects massive amounts of network traffic data, SCOUT can analyze these data offline to identify and cluster malicious flows, aiding security analysts in prioritizing threats and allocating resources efficiently;
- Facilitation of automatic signature generation: Once flows representing distinct attack types are clustered, SCOUT can serve as the initial step for automatic signature generation. Thus, it could enable the creation of precise attack signatures, leading to the early detection of novel threats that traditional methods might miss;
- Denoising malicious traffic: SCOUT can cluster benign traffic separately, essential for accurately isolating malicious activity. This capability is particularly useful for traffic obtained from distributed honeypot systems. In a large-scale deployment across multiple honeypots, SCOUT could effectively filter out benign traffic, allowing for

the identification of coordinated attack patterns that would otherwise be obscured by noise;

- Improvement of existing signatures: The method can enhance the quality of automatically generated signatures, which are often suboptimal [12]:
  - If a signature is too general and captures flows beyond the intended attack, SCOUT will split this traffic into multiple clusters, enabling the generation of more specific signatures. SCOUT could refine existing signatures, potentially leading to a significant reduction in false positives;
  - Conversely, if a signature is too specific, flows tagged by the signature will be grouped with other relevant flows, allowing for the refinement of the signature. As a result, SCOUT's clustering capabilities could improve the generalizability of signatures, thereby enhancing the detection rate of targeted attacks.

5. Matrix Factorization-Based Clustering

Many clustering algorithms are discussed in the literature [35,36]. As the primary goal of our research is to determine the influence of partial supervision, we focus on those that can be viewed as a matrix factorization problem with additional constraints [19]. Among these techniques, NMF emerges as an intuitive clustering method, leveraging constraints to extract meaningful patterns from data across various domains. Notably, NMF has found widespread application due to its ability to capture the latent structures in data effectively. NMF can be extended to semi-supervised learning by modifying the objective function to penalize discrepancies with labeling. This adaptation allows NMF to incorporate labeled data into the clustering process and enhances performance in scenarios where labeled data are available.

5.1. Non-Negative Matrix Factorization

NMF decomposes matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ into the product of two matrices with non-negative elements. Since not every matrix can be represented in such a form, the algorithm seeks a factorization that minimizes the approximation error calculated in the Frobenius norm by solving the following optimization problem:

$$\arg \min_{\mathbf{V},\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{V}\mathbf{H}^T\|_F, \tag{1}$$

where $\mathbf{V} \in \mathbb{R}^{n \times k}_{\geq 0}, \mathbf{H} \in \mathbb{R}^{m \times k}_{\geq 0}$. We refer to the above variant as NMF-$\|\cdot\|_F$.

NMF plays a significant role in machine learning and data analysis. Firstly, it serves as a dimensionality reduction method. The matrix $\mathbf{H}$ contains the coordinates of the analyzed points in the space spanned by the columns in $\mathbf{V}$. Due to its ease of interpretation, in the case of non-negative data (such as pixel matrices representing images), NMF is an attractive alternative to Principal Component Analysis (PCA). Secondly, NMF can serve as a clustering method for the columns in matrix $\mathbf{X}$ [37,38]. As such, it is often compared to the K-means algorithm, which can be formulated as follows:

$$\arg \min_{\mathbf{C},\mathbf{B}} \|\mathbf{X} - \mathbf{C}\mathbf{B}^T\|_F, \tag{2}$$

where $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_k] \in \mathbb{R}^{m \times k}$ is a matrix whose columns are cluster centroids. Matrix $\mathbf{B} \in \mathbb{R}^{k \times n}$ specifies the membership of columns $\mathbf{X}$ to clusters; $\mathbf{B}_{ij} = 1$ if the i-th observation belongs to cluster j, while $\mathbf{B}_{ij} = 0$ otherwise. Thus, NMF can be viewed as a relaxation of the K-means algorithm with the additional assumption of non-negativity of the centroid coordinates. In this context, $\mathbf{H}$ indicates the cluster membership and k is the model parameter denoting the maximum number of clusters.

A well-known variant of the NMF algorithm is the one where the approximation error is measured not in the Frobenius norm but according to the Kullback–Leibler divergence measure, given by

$$d(\mathbf{A}||\mathbf{B}) = \sum_{i,j}\left(\mathbf{A}ij \log \frac{\mathbf{A}ij}{\mathbf{B}ij} - \mathbf{A}ij + \mathbf{B}_{ij}\right), \tag{3}$$

$\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times m}$. In this case, the optimization task takes the form

$$\arg \min_{\mathbf{A},\mathbf{V},\mathbf{H}\geq 0} d(\mathbf{X}||\mathbf{V}\mathbf{H}^T). \tag{4}$$

We denote this class of NMF algorithms as NMF-$d(\cdot|\cdot)$. Just like in the case of approximation error measured in the Frobenius norm, regularization can be incorporated into the problem.

5.2. Semi-Supervised Clustering

Semi-supervised algorithms use both labeled and unlabeled data. In the clustering task, they allow not only grouping based on the similarity of elements, but also based on the relationships between them; for example:

- Objects labeled A should be grouped together;
- An object labeled A and an object labeled B should belong to different clusters.

One of the semi-supervised clustering algorithms is the Semi-supervised non-negative matrix factorization (SSNMF) introduced by Lee et al. [24]. The SSNMF algorithm was further developed by Haddock et al. [23] to incorporate various measures of object similarity.

Lee et al. [24] proposed an NMF extension incorporating selected observations' tagging. Let $\mathbf{Y} \in \mathbb{R}^{l\times m}$ encode the tagging. Let $\mathbf{Y}_{ij} = 1$ if the i-th column belongs to class j. The proposed model is defined by the following optimization problem:

$$\arg \min_{\mathbf{A},\mathbf{V},\mathbf{H}\geq 0} \|\mathbf{W} \odot (\mathbf{X} - \mathbf{V}\mathbf{H}^T)\|_F^2 + \lambda\|\mathbf{L} \odot (\mathbf{Y} - \mathbf{U}\mathbf{H}^T)\|_F^2, \tag{5}$$

where $\odot$ denotes the Hadamard (elementwise) product, $\mathbf{V} \in \mathbb{R}_{\geq 0}^{n\times k}$, $\mathbf{H} \in \mathbb{R}_{\geq 0}^{m\times k}$, $\mathbf{U} \in \mathbb{R}_{\geq 0}^{k\times m}$, $\mathbf{L} \in \{0,1\}^{l\times m}$, and $\lambda > 0$ is a regularization parameter that determines the importance of the supervised term. The above formulations allow the factorization of matrix $\mathbf{X}$, where some data are unknown: $\mathbf{W}_{ij} = 0$ if $\mathbf{X}_{ij}$ is unknown, and 1 otherwise. Minimizing the second term in Problem (5) ensures the alignment of the low-rank representation $\mathbf{H}$ with the tagging. The columns in matrix $\mathbf{L}$ specify whether a given point is tagged, i.e.,

$$\mathbf{L}_{:,j} = \begin{cases} \mathbf{1}_k & \text{label of } \mathbf{x}_j \text{ is known,} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In the above formulation, $\mathbf{1}_k$ denotes a vector of ones of length k.

Haddock et al. [23] compared the clustering effectiveness of (5) with the approximation error measured according to the Kullback–Leibler divergence (3) measure:

$$\arg \min_{\mathbf{A},\mathbf{V},\mathbf{H}\geq 0} d(\mathbf{W} \odot (\mathbf{X}||\mathbf{V}\mathbf{H}^T)) + \lambda\|\mathbf{L} \odot (\mathbf{Y} - \mathbf{U}\mathbf{H}^T)\|_F^2, \tag{7}$$

where $\mathbf{V} \in \mathbb{R}_{\geq 0}^{n\times k}$, $\mathbf{H} \in \mathbb{R}_{\geq 0}^{m\times k}$, $\mathbf{U} \in \mathbb{R}_{\geq 0}^{k\times m}$, $\mathbf{L} \in \{0,1\}^{l\times m}$, and $\lambda > 0$.

Below, we present Algorithm 1 for solving the optimization problem (5) and Algorithm 2 for addressing problem (7), as implemented by Haddock et al. [23].

Algorithm 1 SSNMF-$\|\cdot\|_F$

Input: $\mathbf{X}, \mathbf{W}, \mathbf{Y}, \mathbf{L}, k, \lambda, T$

Initialize $\mathbf{U}, \mathbf{V}, \mathbf{H}$

for $t = 1, 2, \ldots, T$ **do**

$\mathbf{V} \leftarrow \mathbf{V} \odot \frac{[\mathbf{W} \odot \mathbf{X}]\mathbf{H}}{[\mathbf{W} \odot \mathbf{V}\mathbf{H}^T]\mathbf{H}}$

$\mathbf{U} \leftarrow \mathbf{U} \odot \frac{[\mathbf{L} \odot \mathbf{Y}]\mathbf{H}}{[\mathbf{L} \odot \mathbf{U}\mathbf{H}^T]\mathbf{H}}$

$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{V}^T[\mathbf{W} \odot \mathbf{X}] + \lambda \mathbf{U}^T[\mathbf{L} \odot \mathbf{Y}]}{\mathbf{V}^T[\mathbf{W} \odot \mathbf{V}\mathbf{H}^T] + \lambda \mathbf{U}^T[\mathbf{L} \odot \mathbf{U}\mathbf{H}^T]}$

end for

Algorithm 2 SSNMF-$D(\cdot||\cdot)$

Input: $\mathbf{X}, \mathbf{W}, \mathbf{Y}, \mathbf{L}, k, \lambda, T$

Initialize $\mathbf{U}, \mathbf{V}, \mathbf{H}$

for $t = 1, 2, \ldots, T$ **do**

$\mathbf{V} \leftarrow \frac{\mathbf{V}}{\mathbf{W}\mathbf{H}} \odot \left[\frac{(\mathbf{W} \odot \mathbf{X})}{(\mathbf{W} \odot \mathbf{V}\mathbf{H})} \odot \mathbf{W}\right]\mathbf{H}$

$\mathbf{U} \leftarrow \mathbf{U} \odot \frac{(\mathbf{L} \odot \mathbf{Y})\mathbf{H}}{(\mathbf{L} \odot \mathbf{U}\mathbf{H}^T)\mathbf{H}}$

$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{V}^T\left[\frac{(\mathbf{W} \odot \mathbf{X})}{(\mathbf{W} \odot \mathbf{V}\mathbf{H}^T)} \odot \mathbf{W}\right] + 2\lambda \mathbf{U}^T(\mathbf{L} \odot \mathbf{Y})}{\mathbf{V}^T\mathbf{W} + 2\lambda \mathbf{U}^T(\mathbf{L} \odot \mathbf{U}\mathbf{H}^T)}$

end for

6. SCOUT Integration Network Traffic Analysis Cluster

In this section, we present the architecture of the Network Traffic Analysis Cluster (NTAC), which integrates components implementing the SCOUT method, receiving mirrored network traffic from the honeynet. The NTAC consists of four components—IDS, Flow Generator, Feature Extraction and Labeling Module, and SCOUT Module:

1. IDS Module: This module generates labels when a matching signature is found. We propose using Snort 3.0 [39], a free open-source solution that detects intrusions using a rule-based language, combining protocol-, signature-, and anomaly-based inspection methods. The rules specify actions to be taken when a match is found. The alert action generates an alert containing the source port, source IP, destination port, destination IP, protocol, and timestamp. This information is used to match the alert with the corresponding flow data, including the signature ID (sid) and revision number (rev). Snort can send these alerts to a syslog facility, which can then forward them to the Feature Extraction and Labeling Module. This export can be achieved using standard technologies like rsyslog (https://www.rsyslog.com/ (accessed on 27 May 2024), version 8.2406.0) or Filebeat (https://www.elastic.co/beats/filebeat (accessed on 27 May 2024), version 8.14);
2. Flow Generator Module: This module generates flow data using CICFlowMeter 4.0 [40], which can sniff packets in real-time from the network interface and output flow data. Each flow consists of 83 different features. The output can be sent to a file and easily forwarded via Filebeat;
3. Feature Extraction and Labeling Module: This custom module performs three functions—selecting quantitative features, normalizing their values, and correlating flow data with Snort alert data. The correlation is based on the source port, source IP, destination port, destination IP, protocol, and timestamp. This process enriches the flow data with the signature label (defined by the sid and rev);
4. SCOUT Module: The SCOUT module implements semi-supervised clustering logic using the ssmnf library [23]. It includes a wrapper that converts labeled flow data into either numpy arrays or torch tensors, as defined by the ssnmf interface.

The diagram with architecture overview is presented in Figure 4. The proposed architecture is motivated by the ease of implementation and serves as a prototype system.

The components can be distributed across multiple machines, each dedicated to specific tasks. Future work will focus on optimizing this setup to improve scalability and efficiency. For example, the network traffic is currently mirrored to two components, which may introduce some overhead. While Snort 3.0 and CICFlowMeter 4.0 are effective for our prototype, alternatives like Suricata 7.0.6 or NetFlow 9 [31] may offer better performance or additional features. Furthermore, SCOUT's reliance on NMF and SSNMF is a starting point. Future research will explore other semi-supervised clustering algorithms that may offer faster performance and better scalability, helping to select the best-suited algorithm for specific deployment scenarios. By addressing these challenges, we aim to refine the SCOUT method and ensure its viability for deployment in operational cybersecurity environments. Our ongoing and future research will optimize the NTAC architecture to enhance its efficiency and effectiveness in real-time applications.

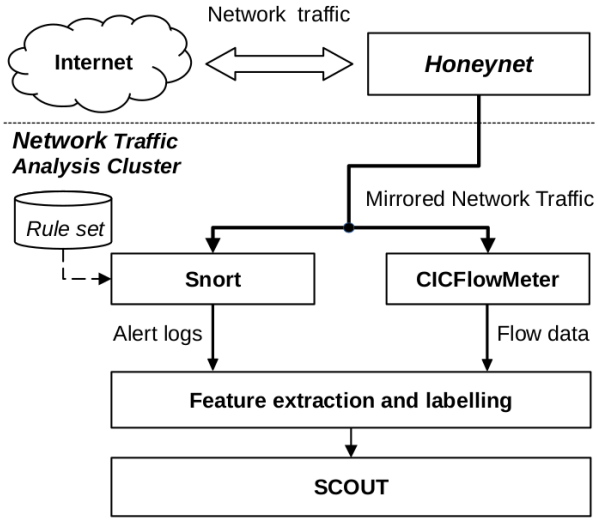


Figure 4. Architecture of Netrwork Traffic Analysis Cluster.

7. Critical Challenges in the SCOUT Methodology

Addressing several critical challenges is essential to ensuring SCOUT's effectiveness in practical applications. This section comprehensively analyses five key factors: scalability, computational complexity, practical limitations, quality of labels, and parameter sensitivity. Understanding and addressing these challenges is crucial for optimizing SCOUT's performance and applicability in real-world network security environments.

7.1. Scalability and Computational Complexity

In the proposed approach, the computational cost of SCOUT is determined by the computational complexity of Non-negative Matrix Factorization (NMF). Although solving NMF problems is NP-hard, many practical algorithms are available. These algorithms often use iterative local optimization schemes that converge to local solutions. While these algorithms do not offer global optimality guarantees, they are widely used in practice. Most iterative NMF algorithms optimize alternately over one variable while keeping the other fixed, a process known as alternating minimization.

In this work, we employed the multiplicative update (MU) rule introduced by Lee et al. [24]. This method is straightforward to implement and does not require parameter tuning, such as step size or learning rate. The ssnmf implementation supports both numpy arrays and torch tensors, the latter of which can benefit from GPU acceleration. The MU method also offers flexibility: it can incorporate both the Frobenius norm and Kullback–Leibler divergence. MU rules can be extended to include supervised terms and weigh observations and labels. The computational complexity of each iteration of the MU algorithm for NMF is $\mathcal{O}(mnk)$, while, for semi-supervised NMF (SSNMF), it is $\mathcal{O}((m+l)nk)$. Since, in general, $m \gg l$, the computational complexity of an SSNMF iteration reduces to $\mathcal{O}(mnk)$ [24].

Despite its advantages, the MU method has some drawbacks. It requires storing all previous samples in matrices at each iteration, necessitating significant memory. To handle large datasets, researchers have explored two main directions:

- Online NMF Algorithms: These algorithms refine the basis matrix with each new data sample without storing past samples [41,42];
- Distributed Algorithms: By partitioning the data and arranging the computations to maximize data locality and parallelism [43,44].

To our knowledge, there has been no integration of online and distributed methods in semi-supervised NMF. We plan to investigate this in future research to enhance SCOUT's scalability and practicality for real-time applications. The primary goal of this research was to examine the influence of labeling, which is why we chose NMF and SSNMF. Future research will focus on optimization and efficiency, exploring some semi-supervised versions of other potentially faster clustering algorithms.

7.2. Sensitiveness to the Quality of Signatures and Parameter Values

The quality of the initial labels influences SCOUT performance. However, the clustering problem, formulated as an optimization problem, offers flexibility. Specifically, reducing the value of the parameter λ can lessen the emphasis on supervised learning, thereby partially addressing the issue of dependency on initial label quality. Additionally, the problem can be reformulated to assign different weights to various labels. Currently, Problems (5) and (7) incorporate knowledge of the given labels using the binary matrix $\mathbf{L}$. However, this problem can be adjusted so that the matrix $\mathbf{L}$ contains real-valued elements, representing the quality of the given labeling. These weights can be determined by expert knowledge about the quality of the signature. This adjustment further addresses the issue of initial label quality.

Estimating the number of clusters, k, which, ideally, corresponds to the number of distinct attack types, is significant. At this moment, this parameter is user-defined and can significantly impact the clustering results. If k is set too low, distinct attacks may not be adequately separated, while setting k too high may lead to the artificial splitting of attack types.

8. Experimental Procedure and Performance Measures

We performed several experiments to evaluate the performance of the SCOUT method comprehensively. The first group of experiments aimed to compare the effectiveness of unsupervised clustering (NMF) and partially supervised clustering (SSNMF). In both cases, two measures of object similarity were compared: Frobenius norm and Kullback–Leibler divergence (3). Moreover, we have benchmarked the performance of SSNMF with frequently employed clustering algorithms, including DBSCAN [45], Agglomerative Clustering [46], and Spectral Clustering [47].

The computation code was implemented in Python 3.9 and executed on a Linux workstation equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz processor (8 cores) and 32 GB of RAM (Dell Vostro 3500, Dell, Round Rock, TX, USA). The code uses scikit-learn 1.4.2 [48] and ssnmf 1.0.3 [10] libraries.

8.1. Data Set, Preprocessing Steps, and Experimental Setup

The selected clustering algorithms were evaluated on data from the CIC-IDS2017 dataset [1]. This dataset contains network traffic records in the form of PCAP files along with metadata obtained using the CICFlowMeter tool. Flow labels correspond to 12 selected attacks (including DoS, DoS Slowhttp, Heartbleed, Brute Force, and scanning) or indicate benign traffic. Flows are characterized by 80 features, including source IP address, source port, destination IP address, destination port, protocol, and communication duration. Additionally, values such as the maximum, minimum, mean, and standard deviation of metrics such as size, number of transmitted packets, or the number of individual TCP flags

are provided. Data were collected for a period of five days, starting from 9 a.m. on Monday, 3 July 2017 , and ending at 5 p.m. on Friday, 7 July 2017.

In the first stage, quantitative features were selected from the CIC-IDS2017 dataset. Subsequently, flows labeled as benign were filtered out. The columns in the data matrix were normalized so that their Euclidean norm equaled 1. We conducted two groups of experiments. In the first group, we compared the NMF and SSNMF algorithms. Each trial benchmarked these algorithms for various parameter settings and shares of labeled data. Figure 5 presents the experimental setup for the comparative study of NMF and SSNMF algorithms. The second experiment compared the unsupervised clustering algorithms DBSCAN, Agglomerative Clustering, and Spectral Clustering. This setup was simpler, as it did not require sampling labels, and we only varied the parameter values.

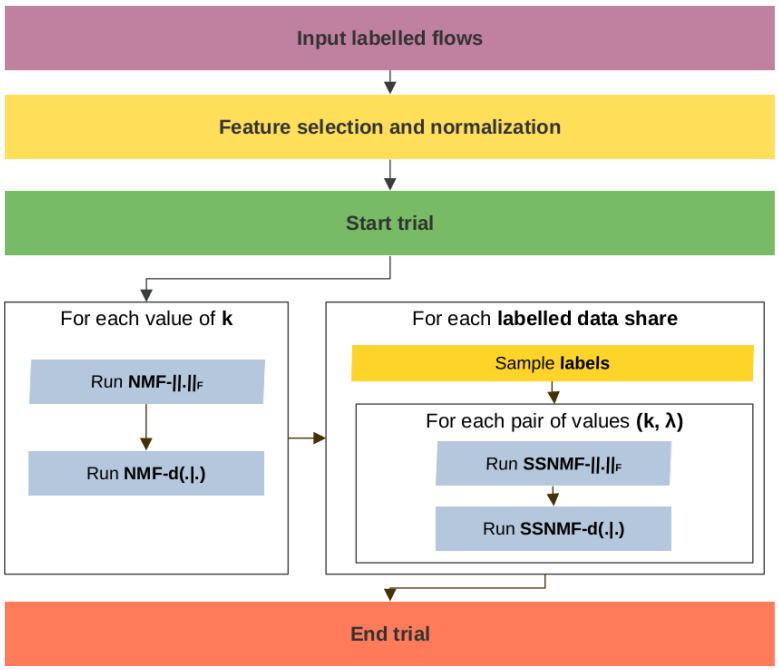


Figure 5. Experimental setup for the comparitive study of NMF and SSNMF algorithms.

8.2. Performance Measurement

Since we compared the clustering algorithms on the labeled dataset, the quality of the solution was assessed by its alignment with the grouping induced by the provided labeling.

One commonly utilized similarity measure for comparing two clusterings of a given set of objects is the Adjusted Rand Index (ARI). Initially defined by Rand, this index quantifies the similarity between two clusterings by considering the proportion of object pairs that are either assigned to the same cluster in both clusterings or different clusters in both clusterings. Formally, in the case of clustering n elements, a being the number of pairs of elements that are in the same subset in both clusterings, and b being the number that are in different subsets in both clusterings, the Rand Index (RI) is given by

$$RI = \frac{a+b}{\binom{n}{2}}. \tag{8}$$

ARI provides a normalized measure that accounts for chance agreement between clusterings, offering a robust assessment of clustering quality:

$$ARI = \frac{RI - \mathbb{E}(RI)}{\max(RI) - \mathbb{E}(RI)}, \tag{9}$$

where $\mathbb{E}(RI)$ denotes the expected value of the Rand Index.

The ARI yields an expected value of zero for independent clusterings and attains its maximum value of 1 for perfectly identical clusterings. This property renders it a robust and interpretable measure for evaluating the effectiveness of clustering algorithms.

9. Experimental Results

We conducted several experiments with various parameter settings to compare the quality of clustering results obtained under partial and no supervision. We aimed to compare the quality of clusters for the parameters at which each algorithm achieved the best results in terms of the ARI (9).

9.1. Parameter Value Selection

The quality of clustering results is highly dependent on the selection of model parameters. Proper parameter tuning can significantly enhance the performance and accuracy of clustering algorithms. Each clustering technique has specific parameters that need to be optimized to achieve the best possible results. Our study focused on fine-tuning these parameters to maximize the Adjusted Rand Index (ARI), a widely used metric for evaluating clustering quality. By systematically varying key parameters and assessing their impact on clustering performance, we aimed to identify the optimal settings that yield the highest ARI values. The subsequent sections detail the parameter selection process for each clustering algorithm employed in our study, including non-negative matrix factorization (NMF), semi-supervised non-negative matrix factorization (SSNMF), DBSCAN, Spectral Clustering, and Agglomerative Clustering.

9.1.1. NMF and SSNMF Parameters

For unsupervised non-negative matrix factorization (NMF), the sole parameter k determines the number of clusters. For semi-supervised non-negative matrix factorization (SSNMF), an additional parameter, λ, governs the relative importance of the supervision term. Intuitively, λ expresses the confidence in the quality of the labels: the larger its value, the more the clustering is influenced by the labels. The parameter k should ideally be close to the expected number of clusters, but, in practical scenarios, this information is often not readily accessible. Therefore, these parameters must be tuned based on some metric, which, in our experiment, is ARI. Future research will evaluate the robustness of the algorithms to different parameter settings.

We compared the quality of results for the parameters where each algorithm achieved the best ARI (9). We tested various combinations of $\lambda \in \{0.001, 0.01, 0.1, 0.5, 1, 10, 100\}$ and $k \in \{5, 10, 20, 30\}$ at several different levels of labeled data. We randomly sampled labeled data according to the label share and ran all algorithms with the given parameter setups. We repeated the process five times for each parameter set and selected the best parameters based on the ARI (9).

The best k parameter for NMF algorithm is shown in Table 1. The selected parameter values are presented in Table 2 for SSNMF-$d(\cdot|\cdot)$ and Table 3 for SSNMF-$\|\cdot\|_F$.

Table 1. Selected parameter values for NMF-$\|\cdot\|_F$ and NMF-$d(\cdot|\cdot)$

Algorithm	NMF-$\|\cdot\|_F$	NMF-$d(\cdot|\cdot)$
k	30	10

Table 2. Selected parameter values for SSNMF-$d(\cdot|\cdot)$.

Label Share	k	λ
0.1	10	0.1
0.3	10	0.1
0.5	10	0.1
0.6	30	10
0.8	10	10

Table 3. Selected parameter values for SSNMF-$\|\cdot\|_F$.

Label Share	k	λ
0.1	5	0.5
0.3	10	0.1
0.5	10	1.0
0.6	20	1.0
0.8	10	1.0

9.1.2. DBSCAN and Spectral and Agglomerative Clustering Parameters

For the DBSCAN algorithm, the parameters tested were the maximum distance between two samples for one to be considered as neighbours ($\epsilon \in \{0.3, 0.5, 1, 10\}$) and the number of neighbours of a point to become a core point (min_samples $\in \{5, 10, 20, 30\}$). For Spectral Clustering and Agglomerative Clustering methods, the number of clusters k varied as follows: $k \in \{5, 9, 10, 11, 15, 20, 30\}$.

The parameters for which each algorithm achieved the best ARI values are DBSCAN with Euclidean distance: $\epsilon = 0.3$, min_samples $= 5$; DBSCAN with Manhattan distance: $\epsilon = 0.5$, min_samples $= 20$; Spectral Clustering: number of clusters $k = 5$; Agglomerative Clustering with Canberra distance: number of clusters $k = 5$; Agglomerative Clustering with Euclidean distance: number of clusters $k = 9$.

9.2. Comparitive Study of NMF and SSNMF Algorithms

In this section, we present a comparative study of the clustering algorithms using the selected parameter values. For unsupervised clustering, we observed ARI (9) values of 0.51 and 0.49 for NMF-$d(\cdot|\cdot)$ and NMF-$\|\cdot\|_F$, respectively (Figure 6). The presented results were obtained for the parameter k, denoting the number of clusters obtained during cross-validation. Table 4 presents the mean ARI values with standard deviations.

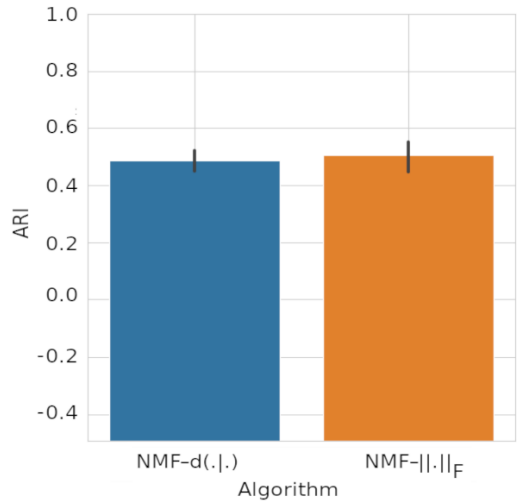


Figure 6. ARI values for unsupervised clustering algorithms.

Table 4. ARI values for unsupervised clustering.

Algorithm	NMF-$\|\cdot\|_F$	NMF-$d(\cdot\|\cdot)$
ARI	0.49 ± 0.05	0.51 ± 0.03

Figure 7 depicts the ARI values for two variants of the SSNMF algorithm: SSNMF-$\|\cdot\|_F$ and SSNMF-$d(\cdot|\cdot)$. In this case, the algorithms had two parameters: k, the target number of clusters, and λ, the regularization parameter. These parameters were set during cross-validation. Notably, regardless of the proportion of labeled data, clustering quality

was consistently higher when the similarity between objects was calculated using Kullback–Leibler divergence (3). Specifically, when labeling half of the data, the SSNMF-$d(\cdot|\cdot)$ algorithm achieved an ARI of 0.67. With 60% of flows labeled, the ARI increased to 0.75. Table 5 presents the mean ARI values with standard deviations for different shares of labeled data.

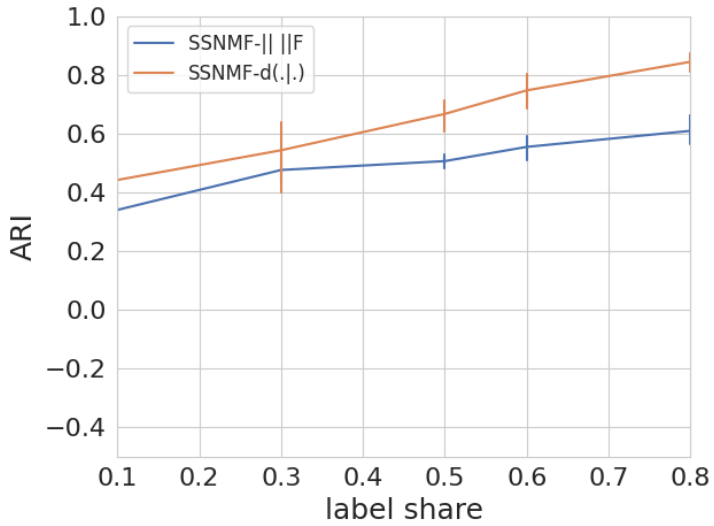


Figure 7. ARI values for semi-supervised clustering algorithms.

Table 5. ARI values' mean and standard deviation for semi-supervised clustering.

Label Share	SSNMF-$\|\cdot\|_F$	SSNMF-$d(\cdot\|\cdot)$
0.1	0.34 ± 0.05	0.44 ± 0.07
0.3	0.48 ± 0.04	0.54 ± 0.16
0.5	0.51 ± 0.03	0.67 ± 0.07
0.6	0.55 ± 0.04	0.75 ± 0.07
0.8	0.61 ± 0.06	0.85 ± 0.04

9.3. Comparative Performance Evaluation with State-of-the-Art Clustering Algorithms

In this subsection, we provide a detailed comparative analysis of the SCOUT method against several state-of-the-art clustering algorithms, including DBSCAN [45], Agglomerative Clustering [46], and Spectral Clustering [47]. This broader set of benchmarks allows for a more comprehensive evaluation of SCOUT's performance beyond the non-negative matrix factorization (NMF) method. For Agglomerative Clustering and DBSCAN, we varied the distance measure. For DBSCAN, we used Manhattan and Euclidean distances, while, for Agglomarative Clustering, we compared Canberra and Euclidean distances.

Figure 8 presents the ARI values for these parameter settings alongside the results for SSNMF-$d(\cdot|\cdot)$ with 0.6 labeled flows and NMF-$d(\cdot|\cdot)$. Agglomerative Clustering with Canberra distace achieved best results in terms of ARI (Table 6). Still, it was lower than ARI of the SSNMF-$d(\cdot|\cdot)$ results (0.75 vs. 0.57).

These results affirm the positive impact of labeling a portion of flows on clustering quality. They underscore the importance of selecting an appropriate similarity measure between objects. Notably, the observed improvement in clustering quality is substantial, indicating the efficacy of incorporating labeled data. Since Agglomerative Clustering produced better results in terms of ARI compared to NMF, this suggests that semi-supervised variants of clustering methods may outperform SSNMF. This hypothesis will be explored in further research.

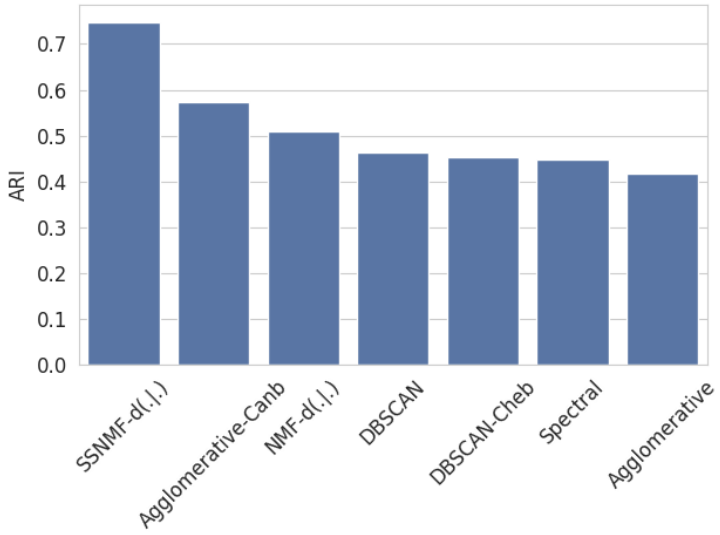


Figure 8. Mean ARI values for SSNMF and unsupervised clustering algorithms.

Table 6. Mean and standard deviation of ARI values for each algorithm

Algorithm	ARI
Agglomerative	0.417 ± 0.001
Agglomerative-Canb	0.574 ± 0.001
DBSCAN	0.463 ± 0.032
DBSCAN-Cheb	0.453 ± 0.031
NMF-d(.|.)	0.510 ± 0.030
SSNMF-d(.|.)	0.748 ± 0.076
Spectral	0.447 ± 0.001

10. Conclusions and Future Directions

This paper introduces a semi-supervised method for clustering malicious network traffic, recognizing that connections with honeypots often indicate known attacks. Our approach groups flows related to specific attacks into dedicated clusters while ensuring that flows resembling other attack signatures are appropriately excluded.

Our method aims to create and refine attack signatures, leveraging semi-supervised algorithms for accurate flow labeling. We advocate for using IDS software for labeling, emphasizing the importance of high-quality signatures from official repositories or expert sources to maintain accuracy. We proposed an architecture of a Network Traffic Analysis Cluster that incorporates SCOUT into the cybersecurity infrastructure.

Through comparative analyses, we demonstrated the effectiveness of our approach in improving clustering results, promising advancements in threat detection capabilities. Future research will validate our approach with real-world data and explore other semi-supervised clustering techniques, including online learning methods, to enhance scalability and robustness.

Author Contributions: Conceptualization, A.K. and E.N.-S.; methodology, A.K. and E.N.-S.; software, A.K.; validation, A.K.; formal analysis, A.K. and E.N.-S.; investigation, A.K.; resources, A.K. and E.N.-S.; data curation, A.K.; writing—original draft preparation, A.K. and E.N.-S.; writing—review and editing, A.K. and E.N.-S.; visualization, A.K.; supervision, E.N.-S.; project administration, E.N.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting this study are publicly available in the Canadian Institute for Cybersecurity repository at https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 24 May 2024)

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Portugal, 22–24 January 2018*; Mori, P., Furnell, S., Camp, O., Eds.; SciTePress: Setúbal, Portugal, 2018; pp. 108–116. [CrossRef]
2. Hwang, R.; Peng, M.; Huang, C.; Lin, P.; Nguyen, V.L. An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access* **2020**, *8*, 30387–30399. [CrossRef]
3. Kruczkowski, M.; Niewiadomska-Szynkiewicz, E. Support Vector Machine for Malware Analysis and Classification. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, 11–14 August 2014—Volume I*; IEEE Computer Society: Washington, DC, USA, 2014; pp. 415–420. [CrossRef]
4. Martínez Garre, J.T.; Gil Pérez, M.; Ruiz-Martínez, A. A novel Machine Learning-based approach for the detection of SSH botnet infection. *Future Gener. Comput. Syst.* **2021**, *115*, 387–396. [CrossRef]
5. Setianto, F.; Tsani, E.; Sadiq, F.; Domalis, G.; Tsakalidis, D.; Kostakos, P. GPT-2C: A GPT-2 parser for Cowrie honeypot logs. *arXiv* **2021**, arXiv:2109.06595.
6. Gómez, G.; Kotzias, P.; Dell'Amico, M.; Bilge, L.; Caballero, J. Unsupervised Detection and Clustering of Malicious TLS Flows. *arXiv* **2021**, arXiv:2109.03878.
7. Diallo, A.F.; Patras, P. Adaptive Clustering-based Malicious Traffic Classification at the Network Edge. In Proceedings of the 40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, 10–13 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–10. [CrossRef]
8. Werner, T.; Fuchs, C.; Gerhards-Padilla, E.; Martini, P. Nebula–generating syntactical network intrusion signatures. In Proceedings of the 4th International Conference on Malicious and Unwanted Software, MALWARE 2009, Montréal, QC, Canada, 13–14 October 2009; IEEE Computer Society: Washington, DC, USA, 2009; pp. 31–38. [CrossRef]
9. Udell, M.; Horn, C.; Zadeh, R.; Boyd, S.P. Generalized Low Rank Models. *Found. Trends Mach. Learn.* **2016**, *9*, 1–118. [CrossRef]
10. Haddock, J.; Kassab, L.; Li, S.; Kryshchenko, A.; Grotheer, R.; Sizikova, E.; Wang, C.; Merkh, T.; Madushani, R.W.M.A.; Ahn, M.; et al. Semi-supervised NMF Models for Topic Modeling in Learning Tasks. *arXiv* **2020**, arXiv:2010.07956.
11. Szynkiewicz, W.; Niewiadomska-Szynkiewicz, E.; Lis, K. Deep Learning of Sensor Data in Cybersecurity of Robotic Systems: Overview and Case Study Results. *Electronics* **2023**, *12*, 4146. [CrossRef]
12. Szynkiewicz, P.; Kozakiewicz, A. Design and evaluation of a system for network threat signatures generation. *J. Comput. Sci.* **2017**, *22*, 187–197. [CrossRef]
13. Skłodowski, J.; Arabas, P. Application of suffix trees to efficient presentation of honeypot registered sessions. *Cybersecur. Law* **2023**, *9*, 298–315. [CrossRef]
14. Satria, E.; Huda, T.P.S.; Iqbal, M.; Sarjana, F.W. The investigation on cowrie honeypot logs in establishing rule signature snort. *IOP Conf. Ser. Earth Environ. Sci.* **2021**, *644*, 012031. [CrossRef]
15. Sadique, F.; Sengupta, S. Analysis of Attacker Behavior in Compromised Hosts During Command and Control. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–7. [CrossRef]
16. Krzysztoń, M.; Lew, M.; Marks, M. NAD: Machine Learning Based Component for Unknown Attack Detection in Network Traffic. In *Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools*; Springer International Publishing: Cham, Switzerland, 2022; pp. 83–102. [CrossRef]
17. Kolodziej, J.; Krzysztoń, M.; Szynkiewicz, P. Anomaly Detection In TCP/IP Networks. In Proceedings of the ECMS 2023 Proceedings, Florence, Italy, 20–23 June 2023; pp. 542–548. [CrossRef]
18. Perdisci, R.; Lee, W.; Feamster, N. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In Proceedings of the Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010.
19. Tropp, J.A. Topics in Sparse Approximation. Ph.D. Thesis, The University of Texas at Austin, Austin, TX, USA, 2004.
20. Kim, J.; Park, H. *Sparse Nonnegative Matrix Factorization for Clustering*; Technical Report; Georgia Institute of Technology: Atlanta, GA, USA, 2008.
21. Ding, C.; He, X.; Simon, H.D. On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. In Proceedings of the 2005 SIAM International Conference on Data Mining (SDM), Newport Beach, CA, USA, 21–23 April 2005; pp. 606–610. [CrossRef]
22. Zhu, X.; Goldberg, A.B. *Introduction to Semi-Supervised Learning*; Synthesis Lectures on Artificial Intelligence and Machine Learning; Morgan & Claypool Publishers: San Rafael, CA, USA, 2009. [CrossRef]
23. Haddock, J.; Kassab, L.; Li, S.; Kryshchenko, A.; Grotheer, R.; Sizikova, E.; Wang, C.; Merkh, T.; Madushani, R.W.M.A.; Ahn, M.; et al. Semi-supervised Nonnegative Matrix Factorization for Document Classification. In Proceedings of the 2021 55th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 31 October–3 November 2021; pp. 1355–1360. [CrossRef]
24. Lee, H.; Yoo, J.; Choi, S. Semi-Supervised Nonnegative Matrix Factorization. *IEEE Signal Process. Lett.* **2010**, *17*, 4–7. [CrossRef]
25. Filipiak, D.; Zapała, A.; Tempczyk, P.; Fensel, A.; Cygan, M. Polite Teacher: Semi-Supervised Instance Segmentation With Mutual Learning and Pseudo-Label Thresholding. *IEEE Access* **2024**, *12*, 37744–37756. [CrossRef]

26. Cai, J.; Hao, J.; Yang, H.; Zhao, X.; Yang, Y. A review on semi-supervised clustering. *Inf. Sci.* **2023**, *632*, 164–200. [CrossRef]
27. Ghadermazi, J.; Shah, A.; Bastian, N.D. Towards Real-time Network Intrusion Detection with Image-based Sequential Packets Representation. *IEEE Trans. Big Data* **2024**, 1–17. [CrossRef]
28. Meng, X.; Wang, Y.; Ma, R.; Luo, H.; Li, X.; Zhang, Y. Packet Representation Learning for Traffic Classification. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'22, New York, NY, USA, 14–18 August 2022; pp. 3546–3554. [CrossRef]
29. Karpowicz, M.P. Adaptive tuning of network traffic policing mechanisms for DDoS attack mitigation systems. *Eur. J. Control* **2021**, *61*, 101–118. [CrossRef]
30. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
31. Claise, B. *Cisco Systems NetFlow Services Export Version 9;* RFC 3954; Cisco: San Jose, CA, USA, 2004. [CrossRef]
32. Claise, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*; RFC 5101; Cisco: San Jose, CA, USA, 2008. [CrossRef]
33. Phaal, P.; Panchen, S.; McKee, N. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*; RFC 3176; RFC Editor: Marina del Rey, CA, USA, 2001. [CrossRef]
34. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
35. Mincer, M.; Niewiadomska-Szynkiewicz, E. Application of social network analysis to the investigation of interpersonal connections. *J. Telecommun. Inf. Technol.* **2012**, *2012*, 83–91. [CrossRef]
36. Taha, K. Semi-supervised and un-supervised clustering: A review and experimental evaluation. *Inf. Syst.* **2023**, *114*, 102178. [CrossRef]
37. Vavasis, S.A. On the Complexity of Nonnegative Matrix Factorization. *SIAM J. Optim.* **2010**, *20*, 1364–1377. [CrossRef]
38. Kim, J.; He, Y.; Park, H. Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *J. Glob. Optim.* **2014**, *58*, 285–319. [CrossRef]
39. Roesch, M. Snort: Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, USA, 7–12 November 1999*; Parter, D.W., Ed.; USENIX: Berkeley, CA, USA, 1999; pp. 229–238.
40. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; A. Ghorbani, A. Characterization of Encrypted and VPN Traffic using Time-related Features. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy—CISSP, INSTICC, Rome, Italy, 19–21 February 2016; SciTePress: Setúbal, Portugal, 2016; pp. 407–414. [CrossRef]
41. Guan, N.; Tao, D.; Luo, Z.; Yuan, B. Online Nonnegative Matrix Factorization With Robust Stochastic Approximation. *IEEE Trans. Neural Networks Learn. Syst.* **2012**, *23*, 1087–1099. [CrossRef] [PubMed]
42. Bucak, S.S.; Gunsel, B. Incremental subspace learning via non-negative matrix factorization. *Pattern Recognit.* **2009**, *42*, 788–797. [CrossRef]
43. Gemulla, R.; Nijkamp, E.; Haas, P.J.; Sismanis, Y. Large-scale matrix factorization with distributed stochastic gradient descent. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'11, New York, NY, USA, 24–27 August 2011; pp. 69–77. [CrossRef]
44. Liu, C.; Yang, H.c.; Fan, J.; He, L.W.; Wang, Y.M. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In Proceedings of the 19th International Conference on World Wide Web, WWW'10, New York, NY, USA, 26–30 April 2010; pp. 681–690. [CrossRef]
45. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.* **2017**, *42*, 1–21. [CrossRef]
46. Fernández, A.; Gómez, S. Versatile linkage: A family of space-conserving strategies for agglomerative hierarchical clustering. *J. Classif.* **2020**, *37*, 584–597. [CrossRef]
47. Damle, A.; Minden, V.; Ying, L. Simple, direct and efficient multi-way spectral clustering. *Inf. Inference A J. IMA* **2018**, *8*, 181–203. [CrossRef]
48. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.