Guidelines and a Framework to Improve the Delivery of Network Intrusion Detection Datasets

Brian Lewandowski[1,2]
[1]*Computer Science, Worcester Polytechnic Institute, 100 Institute Road, Worcester, U.S.A.*
[2]*Raytheon Technologies, 1001 Boston Post Road E, Marlborough, U.S.A.*

Keywords: Network Intrusion Detection, Datasets, Machine Learning, Deep Learning.

Abstract: Applying deep learning techniques to perform network intrusion detection has expanded significantly in recent years. One of the main factors contributing to this expansion is the availability of improved network intrusion detection datasets. Despite recent improvements to these datasets, researchers have found it difficult to effectively compare methodologies across a wide variety of datasets due to the unique features generated as part of the delivered datasets. In addition, it is often difficult to generate new features using a dataset due to the lack of source data or inadequate ground truth labeling information for a given dataset. In this work, we look at network intrusion detection dataset development with a focus on improving the delivery of datasets from a dataset researcher to other downstream researchers. Specifically, we focus on making dataset features reproducible, providing clear labeling criteria, and allowing a clear path for researchers to generate new features. We outline a set of guidelines for achieving these improvements along with providing a publicly available implementation framework that demonstrates the guidelines using an existing network intrusion detection dataset.

1 INTRODUCTION

Network intrusion detection (NID) is a methodology to protect computer networks by analyzing network traffic in order to identify malicious network traffic (Chou and Jiang, 2022). Researchers have begun to leverage machine and deep learning techniques in order to effectively combat the increasingly complex and evolving attacks taking place on networks today (Yang et al., 2022). In order to research and verify the applicability of these data intensive techniques to network intrusion detection systems (NIDS), one must utilize datasets consisting of network scenarios that involve both benign and malicious activity. To support these efforts a growing number of datasets have been developed and analyzed (Chou and Jiang, 2022; Ring et al., 2019; Yang et al., 2022). Despite the great strides made in NID dataset development, researchers have identified limitations which make it challenging to benchmark methods and perform feature engineering (Chou and Jiang, 2022; Ferriyan et al., 2021; Sarhan et al., 2021b; Sarhan et al., 2021c; Sarhan et al., 2020; Sarhan et al., 2021c; Wolsing et al., 2021).

In this work we seek to reduce the impact of limitations that occur as a result of the handoff of NID datasets from a dataset developer to downstream NID researchers. We propose a set of guidelines to help dataset developers overcome handoff limitations and extend the positive impact these datasets can have on downstream researchers. Our focus on the handoff of NID datasets between researchers has not been well explored in current NIDS dataset research. While many of the guidelines are generic in nature, we provide details on how to specifically implement them for NID datasets. In addition to the guidelines, we provide an open source containerized environment and framework to support implementation of the guidelines[1].

Figure 1 shows the dataset development process adapted from descriptions in recent research to show where this work logically fits (Sarhan et al., 2021b; Komisarek et al., 2021). As can be seen highlighted in the figure, we focus on improvements for NIDS dataset feature and label generation which leads to additional improvements for the final delivery of the

This document does not contain technology or Technical Data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

[1]https://github.com/WickedElm/niddff

Lewandowski, B.
Guidelines and a Framework to Improve the Delivery of Network Intrusion Detection Datasets.
DOI: 10.5220/0012052300003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 649-658
ISBN: 978-989-758-666-8; ISSN: 2184-7711

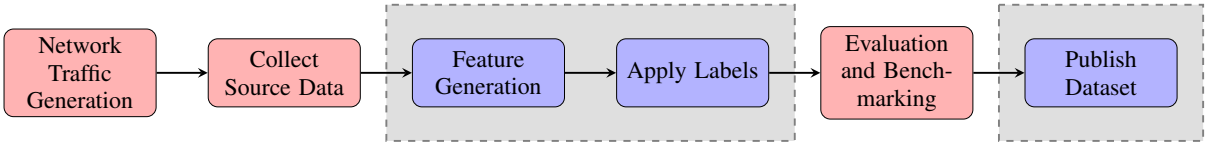

Figure 1: An overview of the NID dataset development process. The areas that the guidelines seek to improve are depicted in blue with a gray background.

dataset. The provided set of guidelines can be used such that the end delivery of a NIDS dataset includes the original source network data as well as concrete scripts for generating each feature and label. With both of these items in hand, researchers will be able to reliably recreate a dataset from source data, ensure the same features are available across multiple datasets, and perform additional feature engineering.

The main contributions of this work are as follows:

- To the best of the authors' knowledge this is the first work to focus specifically on the handoff of NID datasets
- We identify the limitations that affect the handoff of NID datasets between researchers
- Guidelines are provided for overcoming limitations currently present in the NID dataset handoff process between researchers
- An open source containerized environment is provided which contains a standard toolset and a feature engineering framework to support implementation of the guidelines

The remainder of this work is outlined as follows. In Section 2 related works that look to improve the NID dataset development process are explored. Section 3 outlines the limitations related to NID datasets that this work seeks to address. In Section 4 we discuss the details of our proposed guidelines along with the developed containerized environment and framework. We conclude and outline future work in Section 5.

2 RELATED WORK

Publicly available NID datasets that have been developed by researchers are well explored and analyzed in the literature through a number of surveys (Chou and Jiang, 2022; Ring et al., 2019; Yang et al., 2022). These surveys break down the various datasets by different criteria such as data format, real versus synthetic data, availability, as well as statistical data regarding the datasets. For this reason we focus this section on other NID dataset development tools and dedicate Section 3 to discuss work related to NID dataset limitations.

One of the earliest tools concerned with NID datasets was FLAME (Brauckhoff et al., 2008). The main goal of the FLAME tool was to take existing netflow data and augment it by injecting new anomalies into the existing flows. In doing this, it would allow researchers to capture live network traffic and then augment it later with anomalies resulting in a dataset usable for developing NID methodologies.

With a goal similar to FLAME, the ID2T tool also focused on augmenting network data with attacks (Cordero et al., 2015; Cordero et al., 2021). The ID2T tool, however, approached this from the packet level, ingesting packet capture (PCAP) files as opposed to netflow data. In addition, the ID2T tool is capable of providing reports regarding the attacks injected such that they can be used to facilitate data labeling. These qualities allowed ID2T to be capable of injecting a larger variety of network attacks making it useful for the generation of new NID datasets that combine live network traffic with synthetic attacks.

The INSecS-DCS tool (Rajasinghe et al., 2018) is another NID dataset creation tool which has an expanded scope compared to both FLAME and ID2T. Rather than focus on injecting attacks, INSecS-DCS focuses on processing packet data, live or from a PCAP file, such that one can customize the features to include in a final processed dataset. These features can be captured as packet level statistics or based on time windows.

Another related work introduces NDCT (Acosta et al., 2021), which provides a toolkit for the collection and annotation of cybersecurity datasets. NDCT is presented as a system that is primarily used during a cybersecurity scenario exercise. During the scenario execution, users are provided with dialogues to annotate specific packets for tasks such as labeling. These annotations can also be used to generate rules such that similar packets receive the same labeling, making the labeling task more efficient.

Our work is distinguished from these related works in several ways. First, our guidelines and framework do not explicitly focus on injecting attacks into existing source data. Our implementation, however, complements both FLAME and ID2T such that

one could use both tools as part of a pipeline for NID dataset creation within the framework. Similarly, one could incorporate the INSecS-DCS tool for feature creation within our framework. We currently incorporate both Zeek[2] and Argus[3] for our container environment, however, the intention is to grow the environment such that tools such as these can be incorporated. While NDCT focuses on supporting the live annotation of network data during scenario execution, our framework would support downstream feature engineering on the resulting source data. The other main differentiator for our work compared to those discussed here, is that we focus on being able to reproduce a dataset from source files and facilitate its exchange between researchers. The previous works in this space do not generally have this focus as they seek to improve the actual NID data itself as opposed to the process of creating it.

3 NID DATASET LIMITATIONS

3.1 Reproducibility

The main focus of the guidelines and framework presented in our work is to increase the ability of researchers to reproduce datasets from source files. To be clear, we are not concerned with reproducing and re-executing a NID scenario. Rather, we would like to take the resulting PCAP and netflow files from such a scenario and be able to reliably reproduce the dataset's original features and then perform further feature engineering.

An example of the need for this type of reproducibility has been researched recently by analyzing the usage of publicly available datasets by downstream researchers (Chou and Jiang, 2022). In most instances, the original datasets are augmented in some way, however, it was found that most of the work related to these augmentations was unable to be duplicated due to insufficient code, documentation, or both (Chou and Jiang, 2022). The framework delivered in our work seeks to improve this situation by providing a standard way to document and code dataset augmentations from source files.

Other work has proposed a set of content and process requirements for generating a reproducible dataset (Ferriyan et al., 2021). The content requirements outlined include providing full PCAP files with their data payload, anonymization of network traffic, providing ground truth data, using up-to-date network traffic, labeling the data, and providing information regarding encryption. The process requirements pertain to information that should be provided in order to make generating the dataset reproducible. Our guidelines support these requirements, and we look to extend them with our framework through inclusion of the scripts used to generate features and perform labeling, along with full PCAP files. This leaves no ambiguity in descriptions for how to regenerate a dataset.

In addition to these works, there is no shortage of NID literature that discusses the need to have reproducible datasets (Cordero et al., 2021; Lavinia et al., 2020; Sharafaldin et al., 2018a; Kenyon et al., 2020).

3.2 Unclear Labeling Criteria

One of the major challenges that researchers face when working with NID datasets is the lack of datasets with complete and accurate labeling (Lavinia et al., 2020; Moustafa et al., 2019). Many of these labeling issues arise as the task is often performed by human analysis, making it both time-consuming and error-prone (Lavinia et al., 2020). In other cases, the labeling criteria used is either incomplete or influenced negatively by previous errors in the dataset generation process (Lanvin et al., 2022).

For these reasons, accurate labels along with truth data is generally considered a major component of a useful NID dataset (Cordero et al., 2021; Rajasinghe et al., 2018; Komisarek et al., 2021). While ground truth on its own is useful, it can be misleading depending on the granularity in which it is provided. For instance, given only IP addresses and timestamps one would have to assume that any traffic related to that IP address is malicious, however, it is typically the case that a mixture of both benign and attack traffic would be present. For this reason, our proposed guidelines and framework call for the inclusion of labeling scripts which may make use of ground truth data if necessary. We note that in the case of manually labeled data this scripting could simply be index-based using the ordering of the data being processed.

3.3 No Standard Feature Set

In a recent series of papers, Sarhan et al. explore limitations of current datasets and the impact these limitations have on evaluating methods across multiple networks and transitioning research into practical applications (Sarhan et al., 2020; Sarhan et al., 2021a; Sarhan et al., 2021c; Sarhan et al., 2021b). The main limitation explored in these works is the fact that with such varied features included with delivered datasets, one cannot reliably compare a methodology

[2]https://zeek.org/

[3]https://openargus.org/

across multiple networks to test for generalizability. This leads to hindrances during the transition from research to practical applications.

Our work looks to extend the ideas expressed by Sarhan et al. in order to enable researchers to overcome these identified limitations. We aim to make it easier for researchers to provide a dataset that is reproducible from source data and easily expanded or adjusted. In this way, one could easily use a standard feature set as well as research augmenting such a feature set for improvements.

While not the main focus, both (Komisarek et al., 2021) and (Layeghy et al., 2021) discuss and tackle the need to use a common feature set for comparison of their methods as opposed to using proprietary features delivered with most NID datasets. Both works provide informative descriptions regarding the features used in their research. In addition, (Layeghy et al., 2021) provides the actual calculations for the features used in their work. We believe this is a step in the right direction for the level of detail necessary to reproduce datasets from source data. We seek to naturally extend this information into scripts that are provided along with source network captures to make reproducing and extending the dataset more accessible and leave less opportunity for error.

4 NID DATASET DELIVERY GUIDELINES

4.1 The Intrinsic Value in NID Datasets

We believe it is worthwhile to provide a brief discussion regarding the intrinsic value provided by NID datasets as related to their development and subsequent distribution. Namely, the intrinsic value of a NID dataset is created during the scenario development, execution, and *source data* collection and not by the final delivered features. To be clear, the final features are valuable, but they are representative of a separate feature engineering activity that takes place after the intrinsic value of a network scenario has been captured in source data. In other words, the value provided by the NID dataset is derived from the actual network intrusion scenario and its collected source data. A researcher could provide any number of derived features with varying degrees of value for attack detection, however, the intrinsic value of the source data remains constant as it is derived from the scenario that was captured.

One goal of our framework is to highlight these two separate activities by advocating for the delivery of both source data and separate scripts that generate the features that take place during any subsequent feature engineering. Providing both items delivers the value of both activities to downstream researchers.

4.2 Guidelines in Detail

The main ideas behind the proposed guidelines are simple in statement but oftentimes overlooked in practice. Specifically considering the hand off of datasets from one researcher to another; the guidelines focus on ease of access, reproducibility from source data, verification, and extension. The guidelines are meant to provide general guidance for making the delivery of NID datasets meet these four areas of focus and reduce the impact of the limitations discussed in Section 3. We note that our framework allows for the specific implementation of the guidelines to vary depending on the particular methods employed by researchers. While some common tools are provided in our framework environment, we expect it to expand to meet researchers' needs as discussed further in Section 5. In addition, it is important to make the distinction that when we reference reproducibility of a dataset, we refer to reproducing the dataset's final features from the original source data as opposed to recreating and re-executing the dataset's NID scenario.

The ten guidelines are outlined and described in Table 1 along with their justification and details regarding how NID researchers can implement each guideline, with a focus on our companion framework. Guidelines one through four pertain to providing downstream researchers with the resources necessary to actively reproduce and enhance the provided dataset. Guidelines five through nine outline steps that can be taken to ensure that all the dataset features and labels can be regenerated from source data, and that the steps for this generation of features can be verified and understood by downstream researchers. Finally, guideline ten is specifically included to emphasize that the delivered datasets can be considered active projects and adjust over time for any errors found after initial presentation to researchers. This aims to help avoid situations such as with the KDD Cup '99 (kdd, 1999) and CICIDS2017 (Sharafaldin et al., 2018b) datasets, where researchers have found issues with the original datasets resulting in multiple variants of datasets being available with specific corrections (Tavallaee et al., 2009; Lanvin et al., 2022; Engelen et al., 2021).

Table 1: Guidelines for improving the handoff of NID datasets from dataset researchers to downstream researchers.

Guideline	Justification	Implementation Details
(1) Provide direct access to all data and scripts for dataset	The main purpose of this guideline is to prevent barriers to obtaining datasets. (Ring et al., 2019; Cordero et al., 2021)	This can be achieved through a simple download script. The implemented framework provides a mechanism such that dataset developers can provide metadata consisting of a download URL and destination file name to meet this guideline.
(2) Include complete source data to the most detailed extent possible	Full source data is necessary to adequately reproduce and/or augment a dataset. (Ring et al., 2019; Cordero et al., 2021; Ferriyan et al., 2021)	This should generally be a standard format such as PCAP or netflow. Full PCAP files are more favorable than partial PCAP files with no payload. If only netflow data is available, a full collection of attributes is better than a partial collection.
(3) If possible, provide access to all tools needed to generate dataset	Differences in tools, environments, and their versions can limit the ability of downstream researchers to obtain the same results as intended by the original dataset authors. Without this, extending the dataset with feature engineering may not be successful. (Chou and Jiang, 2022; Sarhan et al., 2021c; Cermak et al., 2018)	The implemented framework meets this guideline by providing a containerized environment with specific versions of tools such as Zeek and Argus. This ensures that users of the framework can use the same baseline of tools and environment as was used by the original dataset developers.
(4) Provide documentation indicating how to reproduce a dataset from source data	Clear documentation reduces ambiguity provided in general descriptions of dataset creation. Differences in commands used to generate a dataset from source can produce different results than the original dataset. (Chou and Jiang, 2022; Ferriyan et al., 2021)	Versions of tools and specific commands used to execute them should be documented. The provided framework is self-documenting as researchers can review YAML files for each dataset to view the commands used to generate them as discussed in Section 4.3.
(5) Include source code needed to reproduce dataset features	Providing feature generation source code ensures downstream researchers can duplicate a dataset, verify feature correctness, and understand details of the feature calculation. (Ferriyan et al., 2021; Ring et al., 2019)	One should avoid making code too specific to a particular user environment. The implemented framework supports this guideline with a containerized environment, specific directories for feature generation scripts, and infrastructure to support features generated with network analysis tools.
(6) The source code for each feature should be easily identifiable	This guideline is recommended to make analysis of the features of a dataset more accessible for downstream researchers. (Lanvin et al., 2022; Chou and Jiang, 2022)	This can be implemented through naming conventions for scripts that match the final feature name and techniques such as using a separate script or function for each feature. The implemented framework supports this by enforcing these conventions in its interfaces with network analysis tools.
(7) The generation of each feature should be independent from others	This guideline is recommended to avoid execution dependencies between features and it facilitates the ability to remove or add new features by downstream researchers. This also makes the code for each feature more understandable and reviewable. (Lanvin et al., 2022; Chou and Jiang, 2022)	The implemented framework supports this guideline in the way it interfaces with network analysis tools to generate features in an independent manner where possible. In addition, the built-in framework encourages this by providing standard configuration files that can be used to identify each feature script to run.

Table 1: Guidelines for improving the handoff of NID datasets from dataset researchers to downstream researchers (cont.).

Guideline	Justification	Implementation Details
(8) Apply guidelines outlined for features to labels as well	While labels are significant for model training, during dataset generation time, they can be considered a special case of features. In this way, we want to apply guidelines (4), (5), and (6) to labels as well. (Ferriyan et al., 2021; Lavinia et al., 2020; Cordero et al., 2021; Rajasinghe et al., 2018; Komisarek et al., 2021)	The implemented framework supports this goal by providing the same infrastructure available for feature development to label development.
(9) Make source code for labeling distinct from other features	This guideline is recommended to make the labeling criteria used for a dataset clear for collaborating researchers. Because the label features/procedure can inform machine learning model design decisions it is helpful to have it distinctly identifiable. For example, if the labeling criteria is based on a single IP address, it is likely that the IP address features should not be provided to a model. (Lanvin et al., 2022)	The implemented framework supports this guideline by having a separate step of processing for label scripts and by having them contained in a separate directory for a given dataset.
(10) Provide a mechanism to receive and implement feedback from researchers to correct issues and improve dataset	This guideline encourages collaboration between NID researchers, allows a dataset to remain current, and provides a feedback loop to dataset researchers to correct any issues found by the research community. (Ring et al., 2019; Lanvin et al., 2022)	The implemented framework supports this guideline through its use of scripting and metadata to describe a dataset such that each dataset can be maintained in an independent source code repository or as part of the default environment.

4.3 Framework Details

In this section we cover the main ideas of our containerized environment and implementation of the guidelines. As an example of the implementation, we developed a demo dataset which takes a single PCAP file from the UNSW-NB15 dataset (Moustafa and Slay, 2015) and duplicate most of the original dataset's features and extends them to contain new features. For brevity, many specifics regarding the framework's usage have been omitted. For additional details we recommend consulting the framework repository.

4.3.1 Container Environment

We provide a containerized environment to support our implementation in order to improve reproducibility and eliminate the need to install multiple tools used by other researchers. Currently, this minimal environment includes the Zeek and Argus network analysis tools as well as python[4] and a set of default python libraries as described in the tool's repository. It is expected that this would grow in the future, however, we consider this an adequate starting point to demonstrate its usefulness.

The intention of our framework is that the tool and our container would be used in conjunction together, however, the container environment could be used on its own just to ensure specific versions of tools are easily accessible. Running the container without specifying a command to execute will place the user into a shell prompt with access to the installed tools. The intended method of executing the environment, however, is to map the container's disk drive $/nidd\,f\,f$ to the directory of the user's local repository of our tool infrastructure. This allows for the development of a dataset using the framework and container in a variety of ways.

[4]https://www.python.org/

4.3.2 Framework Implementation

Our implementation provides a standard format for defining and delivering NID datasets using configuration files, naming conventions, and a standard directory structure. At the core of the implementation we read in a YAML configuration file customized for a dataset and use that information to fully process the dataset from source. The high level algorithm followed by the tool can be seen in Algorithm 1.

Algorithm 1: General processing used to generate a NID dataset based on an input configuration file. The input file is processed in a top-down manner with a loop for processing multiple source files prior to combining them together at the end.

Input: *config*, YAML configuration file
Output: *dataset*, NID dataset suitable for ML

Read in *config*

Store documentation information from *config*
Process setup options

Read in metadata for source data
if *download_source* == TRUE **then**
    Download all source PCAP and Netflow files
end if

for each source file **do**
    Execute feature processing commands
    Execute label processing commands
    Execute post-processing commands
    Save intermediary dataset file
end for

Execute final dataset processing commands
Combine intermediary dataset files

return *dataset*

4.3.3 Dataset Directory Structure

Each NID dataset has its configuration and generation scripts contained in a dedicated directory. This allows it to be maintained by the original dataset developers and then plugged into the framework by consumers of the dataset. The general structure of a dataset directory is shown in Figure 2 where one can see the YAML configuration file, directories for source metadata, ground truth metadata, output files, and each processing step's files.

For the source and ground truth data, the directory contains metadata files which are in a comma-separated format where each line contains a download URL and the destination file name which is read in by the framework when acquiring source data. In addition, each processing step can contain simple files with the naming convention $\_\_load\_\_.<tool>$ where $<tool>$ is one of the framework's supported tools such as Zeek or Argus. While the particulars of how each tool behaves varies, these files have each line denote a single feature or process to run for a given tool. If applicable, an associated script with the same name as the feature it generates is contained in the same directory. In other words, users can easily identify the features being generated by reviewing the $\_\_load\_\_.<tool>$ files and the scripts that they reference. This promotes having easy to identify source code for each feature as indicated in guideline six, as well as having self-contained features as indicated in guideline seven.

```
dataset/
├── config.yaml
├── source/
│   └── pcaps.meta
├── ground_truth/
│   └── gt.meta
├── output/
├── step_acquire_source_data/
│   ├── __load__.argus
│   ├── __load__.python
│   └── __load__.zeek
├── step_feature_processing/
│   ├── __load__.argus
│   ├── __load__.python
│   └── __load__.zeek
├── step_label_processing/
│   ├── __load__.argus
│   ├── __load__.python
│   └── __load__.zeek
├── step_post_processing/
│   ├── __load__.argus
│   ├── __load__.python
│   └── __load__.zeek
└── step_final_dataset_processing/
    ├── __load__.argus
    ├── __load__.python
    └── __load__.zeek
```

Figure 2: A default directory structure for a dataset within the proposed framework. Each processing step has its own directory intended to contain loading scripts for supported tools as well as any other scripts used in a given step. It should be noted that these directories are only needed if they are used for a given dataset. For instance, the framework takes care of default processing for several stages but the user has the option of customizing each stage with their own scripts.

4.3.4 Dataset Configuration File

Each dataset has a YAML configuration file that drives its creation. As seen in Listing 1 it contains documentation, options, and can contain a mix of built-in framework commands as well as custom commands to execute. For example, the framework takes information from the *setup_options* section and determines what source files to download during the *step_acquire_source_data* step. Other built-in commands such as *run_zeek* have default behavior requiring little setup on the user's part in the configuration file. In general, these commands look into the current step's directory and reads an associated $\_\_load\_\_.<tool>$ file. This file is then used by the framework to either generate features, labels, or perform some other intermediary processing. Aside from commands supported by the framework, user's can also specify any custom commands or scripting to execute, and they will be processed in the order they appear in the file. For these commands, users have access to a number of built-in variables that can be accessed in order to direct particulars such as paths to source files to read in and where to place output. The main benefit of this single configuration file is that it fully self-describes how the dataset is created and provides the information needed for users to access the code used to generate features and perform labeling.

4.3.5 Benefits for NID Dataset Developers

The framework implementation provides several benefits for NID dataset developers. First, it provides enough flexibility such that there are varying degrees of buy-in for using the framework. For instance, suppose a NID dataset researcher only provides source files and ground truth data or has a previously generated dataset that they would like to incorporate into the framework with little effort. This can be achieved through the framework by generating the source file metadata files and ground truth metadata files. While minimum effort is required by the NID dataset researcher, it provides additional accessibility of the files to downstream consumers. On the other end of the spectrum, the container environment provides tools for analyzing source data which can be taken advantage of by NID dataset researchers. This use of the container allows downstream researchers to use the same versions of the software when working with the dataset.

Another benefit for NID dataset researchers is that the framework implementation provides an organized structure to follow and self-documents how the dataset features and labels were generated from source data. When updating the dataset or expanding it, the change history of the configuration files within the framework can be inspected to track the changes provided there are no updates to the source data. Additionally, any improvements or feedback can be provided from end users back to the NID dataset researcher by lightweight updates to these configuration files.

The intent of this framework is such that no significant additional work is imposed on NID dataset developers as all the steps it encapsulates must already be performed to generate a given dataset. The emphasis of the framework and guidelines is such that these steps are simply organized in a standardized manner.

```
documentation:
  niddff: niddff/niddff:0.1

setup_options:
  dataset_name: demo_dataset
  source_data: unsw-nb15
  ground_truth_data: unsw-nb15
  clean_output_directory: True
  expected_outputs:
    - unsw_nb15_dataset.csv

  argus:
    clean: True
    arguments: -S 60 -m
    execute_ra: True

step_acquire_source_data:
  download: True

step_feature_processing:
  - run_zeek
  - run_argus
  - run_python_scripts

step_label_processing:
  - run_python_scripts

step_post_processing:
  - run_combine_features

step_final_dataset_processing:
  - run_combine_data
```

Listing 1: A sample input file consumed by our framework specifying where to obtain source data and how to process it to produce a final dataset. Options can be overridden on the command line if necessary.

4.3.6 Benefits for NID Dataset Consumers

This framework also provides benefits for downstream researchers using NID datasets. For researchers looking to simply use the original dataset as provided, there is generally no changes in workflow imposed by the framework though they would

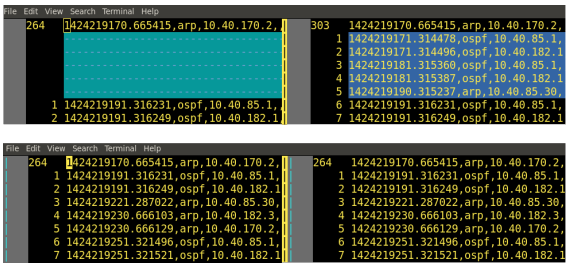

Figure 3: A diff comparison of extracted Argus features from the first PCAP of the UNSW-NB15 dataset. On the top, the left hand side of the diff shows a portion of the original Argus features from the original dataset while the right shows the same section of the output but generated by running Argus with no command line options on the source PCAP. On the bottom, the left hand side of the diff shows the same portion of the original Argus features from the original dataset while the right now shows the same section of the output generated by running Argus with the $-S\ 60$ option. The differences on the top demonstrate the necessity of having the exact command line options used to generate dataset features in order to make a dataset reproducible.

be able to easily obtain the dataset using the download metadata. For researchers seeking to analyze a dataset, the container environment and configuration files approach provides a way for them to reproduce the dataset reliably since all the tools and the command line options used to run them are contained within the scripts. As an example of this benefit, we look at the implementation of our demo dataset, which uses a single PCAP from the UNSW-NB15 dataset (Moustafa and Slay, 2015). As depicted in Figure 3, without using a particular set of options for Argus, one would receive results with an additional 2,529 rows compared to what the original dataset authors intended. This was found experimentally for our research but shows the value of the ambiguity that is removed when researchers have the full commands readily available. Similar benefits are gained by having the full labeling criteria laid out in the dataset configuration files.

An additional benefit comes in the form of being able to generate a standard feature set from any source data. If some standard feature set is not included by the original dataset authors, a researcher can easily adapt the original dataset with a standard feature set in order to facilitate comparisons across multiple datasets. By following the guidelines and using the framework, the scripts to produce such a feature set become plug-n-play for any dataset that uses the same source format.

Similar to this plug-n-play nature of scripts when using the containerized environment and framework, a similar benefit can be realized for individual features of a dataset. As an example, one can consider the situation where two researchers are using the same container version and source dataset and perform different feature engineering. The use of the container and framework allows them to exchange their feature scripts or just the resulting data for *individual features* and simply merge the results into their work. As outlined in Section 5, this ability provides additional benefits if the environment is expanded to include a server-based component.

5 CONCLUSION AND FUTURE WORK

In this work we propose a set of ten guidelines that will improve the handoff of NID datasets between researchers. The focus of these guidelines is to improve ease of access, reproducibility from source data, verification, and the extension of datasets. We believe that considering these areas while generating new datasets will benefit both dataset developers and downstream researchers using the datasets. The provided framework demonstrates these goals and their associated benefits.

While these guidelines are a step forward in progress in this area of research, it does not eliminate all the complexities faced by researchers who want to extend NID datasets. In future work we aim to remove many of these additional complexities by including server-based methods to facilitate these guidelines. With the availability of a server environment, researchers could either use the container environment locally or interact with the server to perform scripting while leveraging the same container environment in both contexts. In this approach, the server could store source data locally eliminating the need to download anything but a final feature set. Additionally, if other researchers had already created a feature on the server, the scripting and data has the potential to be re-used without the need to regenerate anything. This future work would be able to leverage the framework developed here making it a significant step towards even more efficiency gains.

ACKNOWLEDGMENTS

We would like to acknowledge Professor Randy Paffenroth from Worcester Polytechnic Institute for his valuable insights and guidance which helped shape this work.

REFERENCES

(1999). Kdd cup 99. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed: 2022-08-08.

Acosta, J. C., Medina, S., Ellis, J., Clarke, L., Rivas, V., and Newcomb, A. (2021). Network data curation toolkit: Cybersecurity data collection, aided-labeling, and rule generation. In *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, pages 849–854.

Brauckhoff, D., Wagner, A., and May, M. (2008). Flame: A flow-level anomaly modeling engine. In *CSET*.

Cermak, M., Jirsik, T., Velan, P., Komarkova, J., Spacek, S., Drasar, M., and Plesnik, T. (2018). Towards provable network traffic measurement and analysis via semi-labeled trace datasets. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–8.

Chou, D. and Jiang, M. (2022). A survey on data-driven network intrusion detection. *ACM Computing Surveys*, 54(9):1–36.

Cordero, C. G., Vasilomanolakis, E., Milanov, N., Koch, C., Hausheer, D., and Mühlhäuser, M. (2015). Id2t: A diy dataset creation toolkit for intrusion detection systems. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 739–740.

Cordero, C. G., Vasilomanolakis, E., Wainakh, A., Mühlhäuser, M., and Nadjm-Tehrani, S. (2021). On generating network traffic datasets with synthetic attacks for intrusion detection. *ACM Trans. Priv. Secur.*, 24(2).

Engelen, G., Rimmer, V., and Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12.

Ferriyan, A., Thamrin, A. H., Takeda, K., and Murai, J. (2021). Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences*, 11(17).

Kenyon, A., Deka, L., and Elizondo, D. (2020). Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security*, 99:102022.

Komisarek, M., Pawlicki, M., Kozik, R., Hołubowicz, W., and Choraś, M. (2021). How to effectively collect and process network data for intrusion detection? *Entropy*, 23(11).

Lanvin, M., Gimenez, P.-F., Han, Y., Majorczyk, F., Mé, L., and Totel, E. (2022). Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In *CRiSIS 2022 - International Conference on Risks and Security of Internet and Systems*, pages 1–16, Sousse, Tunisia.

Lavinia, Y., Durairajan, R., Rejaie, R., and Willinger, W. (2020). Challenges in using ml for networking research: How to label if you must. In *Proceedings of the Workshop on Network Meets AI & ML*, NetAI '20, page 21–27, New York, NY, USA. Association for Computing Machinery.

Layeghy, S., Gallagher, M., and Portmann, M. (2021). Benchmarking the benchmark – analysis of synthetic nids datasets.

Moustafa, N., Hu, J., and Slay, J. (2019). A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55.

Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6.

Rajasinghe, N., Samarabandu, J., and Wang, X. (2018). Insecs-dcs: A highly customizable network intrusion dataset creation framework. In *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, pages 1–4.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2020). Netflow datasets for machine learning-based network intrusion detection systems. In *Big Data Technologies and Applications*, pages 117–135. Springer.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2021a). A cyber threat intelligence sharing scheme based on federated learning for network intrusion detection.

Sarhan, M., Layeghy, S., and Portmann, M. (2021b). Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection.

Sarhan, M., Layeghy, S., and Portmann, M. (2021c). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1):357–370.

Sharafaldin, I., Gharib, A., Lashkari, A. H., and Ghorbani, A. A. (2018a). Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2018(1):177–200.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee.

Wolsing, K., Wagner, E., Saillard, A., and Henze, M. (2021). Ipal: Breaking up silos of protocol-dependent and domain-specific industrial intrusion detection systems.

Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y., and Han, H. (2022). A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, page 102675.