Journal of Database Management
Volume 31 • Issue 1 • January-March 2020

A Service Architecture Using Machine Learning to Contextualize Anomaly Detection

Brandon Laughlin, University of Ontario Institute of Technology, Oshawa, Canada
Karthik Sankaranarayanan, University of Ontario Institute of Technology, Oshawa, Canada
Khalil El-Khatib, Ontario Tech University, Oshawa, Canada

ABSTRACT

This article introduces a service that helps provide context and an explanation for the outlier score given to any network flow record selected by the analyst. The authors propose a service architecture for the delivery of contextual information related to network flow records. The service constructs a set of contexts for the record using features including the host addresses, the application in use and the time of the event. For each context the service will find the nearest neighbors of the record, analyze the feature distributions and run the set through an ensemble of unsupervised outlier detection algorithms. By viewing the records in shifting perspectives one can get a better understanding as to which ways the record can be considered an anomaly. To take advantage of the power of visualizations the authors demonstrate an example implementation of the proposed service architecture using a linked visualization dashboard that can be used to compare the outputs.

KEYWORDS

Context, Explanation, Intrusion Detection System, Network Flows, Outlier

INTRODUCTION

Monitoring network flows (NetFlows) is an essential part of securing networks. This has become increasingly difficult as the amount of traffic being generated has outgrown the ability to effectively analyze them (Cisco Systems, 2018). In addition to the increasing scale, network data is coming in at faster rates and there is a larger variety of data sources to deal with (Habeeb et al., 2019). With such a large influx of information, analysts are not able to identify threats in a timely manner leading to exploits persisting on networks and only discovered once damage has already been done (Secureworks, 2018). This places an increasing importance on more automated methods such as network intrusion detection systems (NIDS). Existing work on NIDS can be placed into two main categories: signature based, and anomaly based. Signature detection is based on existing attack knowledge using specific criterion for threat detection (Fernandes, Rodrigues, Carvalho, Al-Muhtadi, & Proença, 2019). Anomaly detection establishes baselines and looks for activity that appear as outliers. With the scale of modern big data, security analysts are facing difficulties in reviewing all of the network flows determined as threats by the NIDS (Cisco Systems, 2018).

DOI: 10.4018/JDM.2020010104

Compared to finding anomalies in other applications, the analysis of network security data adds additional challenges. There is a wide range of contexts that will influence whether something is anomalous and if it is anomalous, whether or not it is due to malicious activity. For example, the specific user, their role in the organization, the device in use, the application being used or even the time of day are important considerations for analyzing an outlier. With the increased accessibility and reduced cost of computing power in recent years, machine learning (ML) has increasingly become a tool used to address these challenges (Buczak & Guven, 2016).

Over time, these ML techniques have become very complex to the point where only experts of the system can understand how the system works. It is important to have a clear explanation as to how a certain anomaly rating was generated with more context than just an outlier score as output. Without a proper understanding of the underlying properties used to produce the output it is difficult for an analyst to translate the resulting outlier scores into information that can be acted upon. This is challenging as advanced ML algorithms such as deep learning act as a black box that provide little to no justification as to the results of the classifier (Wang & Siau, 2019).

Most research in machine learning for cybersecurity has been done using supervised learning in which labels are included in the data that identify attacks within the dataset (Buczak & Guven, 2016). While supervised approaches reduce the number of false positives, the dependence on labels is a large limitation (Sommer & Paxson, 2010). As the threats facing networks change very fast, even new datasets become irrelevant quickly as adversaries adjust strategies to avoid detection. Developing labeled datasets can also be very expensive and time consuming and is not very scalable. Training machine learning models without labels in an unsupervised setting can remove these limitations; however, they bring their own set of challenges. One of the largest challenges is the validation of the system (Sommer & Paxson, 2010). By building ML models without labels there is no direct method to assess the accuracy. Not only does this make comparing ML models and choosing the best one very difficult, without an effective validation method, building an easy to understand model is even more difficult.

Network security specialists have extensive background knowledge in protecting networks and a general sense of normal network conditions. This makes them suitable candidates for managing an Intrusion Detection System (IDS) although they may not have the experience needed to setup the ML aspects. There are many steps involved in the data science pipeline including data preprocessing, feature engineering, model selection and tuning model hyper parameters (Zimek et al., 2014). While experienced data scientists may be comfortable configuring these options, many security experts do not have the prerequisite skills (Sacha et al., 2017). One option to integrate the ML components is to employ data scientists to help develop the IDS. However, if the security experts using the system had limited involvement in its development then they may have trouble interpreting the ML results. One needs to abstract the details of the ML operation without losing the ability to extract actionable intelligence from the results. Pure automation would remove the analyst and produce non-optimal results, while depending on the analyst to develop the system would be very difficult and time consuming. It is crucial to balance between abstracting the ML operations behind the scenes and being able to interpret the results.

Whenever a record has been identified as anomalous by a NIDS there is the challenge of determining whether this activity is malicious and how to translate this into something actionable that a network operator can respond to. To help reduce the complexity and time needed to handle a response, it is helpful to have the system provide extra context with an alert. This context can come from baseline data provided alongside the alert to enable exploratory data analysis. This additional context for each alert can allow for better threat intelligence. Instead of relying completely on the analyst to find anomalies or using a ML classifier without context, the authors propose a service architecture that will combine both approaches to take advantage of the scale of ML with the expertise of a human analyst.

The proposed service helps provide context and an explanation for the outlier score given to any network flow record selected by the analyst. The service constructs a set of contexts for the record using features including the host addresses, the application in use and the time of the event. For each context the service will find the nearest neighbors of the record, analyze the feature distributions and run the set through an ensemble of unsupervised outlier detection algorithms. By viewing the records in shifting perspectives one can get a better understanding as to which ways the record can be considered an anomaly.

In addition to providing more context, another way of increasing an analyst's understanding of a ML system is through the use of visualizations. Used as a tool for exploratory data analysis, visualizations can help one better understand and compare the various machine learning approaches in use for improved situational awareness. This situational awareness can help analysts more effectively review alerts and handle false positives by encouraging exploratory data analysis that can help an analyst observe patterns in the data and discover patterns that they do not yet know to look for (Shiravi et al., 2012). This will not only help build better outlier ensembles but also understand anomalies.

To take advantage of the power of visualizations the authors demonstrate an example implementation of the proposed service architecture using a linked visualization dashboard that can be used to compare the outputs. This will provide the opportunity for exploratory data analysis to aid with the interpretation of the network flow records. By providing extra context to individual alerts, analysts can improve on the speed at which they can review network threats. In summary, the contribution of this paper includes:

1. A service architecture for the delivery of contextual information related to network flow records.
2. An example implementation of the service in the form of a web dashboard that provides interactive views to help explore the generated results.

RELATED WORK

Network Security

IT security data has specific characteristics that make the visualization and analysis of this data different than regular business analytics. Factors such as the relationship between source/destination IP addresses and ports makes analyzing this data more challenging as there are often many correlated data elements. These dependencies are an important consideration when combining data such as packet capture and Intrusion Detection System (IDS) alerts. Additionally, time plays a very important role in the data. Another popular domain for identifying anomalous behavior besides network security is financial fraud. Other works have developed systems that have been used to extract logs from different database systems and analyzed them with rulesets to find outliers representing suspicious transactions (Khanuja & Adane, 2018). The service proposed in this work is similar in that it identifies anomalous activity, but this service uses dimension reduction visualizations to generalize unintuitive network features into clearly separable regions in the feature space.

Designing an analysis tool for cybersecurity data presents many challenges such as the volume and variety of data, the complexity of network security data, providing meaningful context to data and ease of navigation. All of these challenges prevent the widespread adoption for analyzing IT security data and need to be carefully considered when designing any tool. The complexity of information systems plays a large role in the challenge of network security data. Related work has built a framework for managing this complexity within information systems (Kaul et al., 2017). The authors present a framework which provides guidance on how to address complexity within the design of information systems. This work builds on these ideas by incorporating dimension reduction visualizations into the service to help mitigate the effects of overwhelming complexity. When using network security tools, not all of the relevant information can be displayed at once. Dealing with big data means that some details are going to be lost when summarizing and aggregating the data for the purposes of analysis. This makes deeper inspection of data difficult. They need to be able to provide meaningful representations of large-scale data at a high level overview while providing details on demand without losing much information.

Cybersecurity data can come from many different sources such as hosts, applications, security appliances and networking devices. These sources may have linked commonalities between them such as matching source and destination IP addresses, ports, protocols or timestamps. These relations may exist between the various data sources such as netflows and IDS alerts. It is important to ensure that information is displayed in ways that clearly illustrate these connections. In addition to the number of sources they will have varying formats, and some will have timestamps associated with them while others will not. Linking the various logs such as netflows with security alerts in a clear way across time is an important function for a tool to have (Best, Endert, & Kidwell, 2014). This is especially true for complex attacks such as advanced persistent threats (APT) that oftentimes only exhibit malicious behavior after a period of time and over periodic intervals. Studies on post deployment changes to IT systems have found that even in the case of using software that is custom developed, the software often suffers from misuse and a mismatch between the intended use case and the abilities of the tool (Kang, 2007). It is for this reason that we emphasize the use of integrating a visual analytics dashboard into the service to enable continuous post-deployment communication between the anomaly detection responders and management.

Machine Learning

Ensembles

Due to the challenges and limitations of supervised machine learning, more research is needed in understanding unsupervised machine learning models. To help avoid model overfitting one can combine several models into an ensemble (Aggarwal & Sathe, 2017). These techniques aim to cancel out the variance in results by combining different machine learning models. There are two main categories of techniques used to improve accuracy with ensembles (Aggarwal & Sathe, 2017). The first set of approaches aims to reduce the bias of the result such as using a sequential set of models in a boosting method. However, without a labeled dataset to validate with this is very difficult and so is seldom used in unsupervised work. The second approach is variance reduction techniques that aim to cancel out the variance in results by combining different machine learning models. This work is in the unsupervised domain and so variance reduction is the approach being focused on. There are many different ways to combine models into an ensemble. Approaches using the same base model include feature subspaces (Keller, Muller & Bohm, 2012), data subsampling (Zimek et al., 2013), and using different model parameters (Schubert et al., 2014). A different category of approaches uses different machine learning types and combines them into a unified score (Nguyen et al., 2010). A recent example of work combining heterogeneous models developed a system that combines matrix decomposition, a replicator neural network and a density-based approach (Veeramachaneni et al., 2016). The service proposed in this work is similar in that it combines heterogeneous models with the added contribution of developing a visualization dashboard to better interpret results.

While ensembles have the potential to be a more versatile solution, the complexity of any system using them is greatly increased. Additionally, ensembles work best when there is a wide variety of models included as a group of very similar models will produce results with a high bias. This means it is imperative that there is a way of assessing the diversity of the models. Other challenges include the best way to normalize and combine the scores of each machine learning model that is a component of the ensemble (Kriegel et al., 2011). Without labeled data assessing these tradeoffs, it is very challenging to accomplish and an important reason for research on unsupervised outlier ensembles receiving less attention until recently (Zimek et al., 2014). Due to the growing importance of unsupervised learning approaches, this work focuses entirely on producing meaningful explanations without any required labeling. While producing labeled data for supervised learning is time consuming some frameworks have been proposed that can reduce this time by querying analysts for labels that will best improve the system (Beaugnon et al., 2017). With some labeled data available some works have devised frameworks combining both supervised and unsupervised approaches (Micenková et al., 2014) and incorporated a human in the loop system to iteratively improve the models as labels become available (Veeramachaneni et al., 2016).

The author's novel contribution in this work is the ability to provide more context to the alerts. Instead of only receiving an outlier score, the service provides more context such as specific attribute subsets that were anomalous. This contextual score combined with visual analytics shows the local context of surrounding records and encourages exploration of the relationships of various network flows. This is essential as the cost of errors for machine learning in network security has a lot of more serious consequences compared to other application domains (Sommer & Paxson, 2010). This context can come from baseline data provided alongside the alert to enable exploratory data analysis. This additional context for each alert can allow for better threat intelligence. The proposed service helps provide context and an explanation for the outlier score given to any network flow selected by the analyst.

Interpretability Challenges

There exist many works on using visual analytics to interact with ML systems (Endert et al., 2017). One of the largest challenges visual analytics aims to address is the interpretation of complex classifiers. Approaches include ways to approximate decision criteria (Ribeiro et al., 2016) and evaluate differences in class assignments (Ren et al., 2017). Examples particular to ensembles includes a linked visualization dashboard that displays various metrics in a scatterplot that graphs each machine learning model (Schneider et al., 2017) and a visualization tool that shows each component decision trees of a random forest for malware detection (Angelini et al., 2017). However most existing works have been done in a semi or completely supervised setting by using record labels as a means of assessing model performances. Visual analytics systems can help users understand machine learning classifier sand improve classifier performance through methods such as explanatory debugging (Kulesza et al., 2015). Trying to troubleshoot the reasons for a lack of model robustness is complicated because of poor model interpretability. Modern machine learning algorithms essentially act as a black box with seemingly random reasoning as to the decision of the model (Wang & Siau, 2019). For this reason, it can be hard to determine how a change to the input will influence the output. Recently there has been a focus on creating methods of better explaining machine learning systems (Mittelstadt et al., 2019).

When dealing with black box models there are two categories of approaches that can be used to provide interpretability: model specific and model agnostic. Model specific methods only work for a specific type of machine learning model. For example, a visualization of the trees in a random forest (Angelini et al., 2017) or the hidden layers of a neural network (Strobelt et al., 2017) are approaches that can only work for those specific model types. Model agnostic approaches can be applied to any machine learning model. They treat the classifier as a black box and only require an output score from the model to operate. The approach used in this work is model-agnostic to allow for greater applicability of the framework. Interpretability for a model can also be defined as a global or local process. A global explanation provides details of how the system works as a high-level description to generate all of the results. Local scope explanations provide the context about a decision relating to specific subsets or individual instances of the dataset such as LIME (Riberio et al., 2016). The service proposed in this work provides local explanations to provide context specific to certain network flow feature subsets. It provides explanations to NIDS using unsupervised learning where no labels are available.

Cybersecurity Visualizations

For visualization tools to be effective, they must provide the user with details on the visualization. Having to simply trust that the visualization works can be a problem. When providing overviews of data by aggregating, the details are lost. When this is done, security analysts often feel that they are losing valuable information in the details. For this reason, analysts often distrust visualizations because they cannot verify exactly how or why the data is represented the way it is (Fink et al., 2009). This is why visualizations need to be able to extract the details from the data while still being clear and consistent. For analysts to trust the visualizations, they need to be able to verify the source of their visuals. An effective program should provide a means of providing explanations to what patterns are being displayed. Compared with command line tools that can pipe data to other applications, visualizations are usually standalone programs (Fernandes et al., 2019). When visualizations support only a specific data format, it is difficult to work with these programs in conjunction with other analytic tools. A visualization tool that can integrate data from both a variety of sources and of differing formats offers greater flexibility in the range of scenarios it could be used for.

Visual analytics involves the combination of interactive visualizations with machine learning to build systems that enable better understanding of data (Keim et al., 2008). Network analysis is a type of analytical problem that can effectively make use of automatic analysis in addition to visualization, making it a great candidate for visual analytics (Keim et al., 2010). To support exploratory data analysis, visualizations can be used to offer increased data density and present large amounts of data in a summarized form (Shiravi et al., 2012). There has been extensive research in the area of visualization for anomaly detection and situational awareness (Shiravi et al., 2012). Even though the use of visualization has been shown to be beneficial for network security data (Goodall, 2009), very few analysts use such tools. Reasons for this lack of adoption include limited interoperability with other tools and having a very limited scope of application. Most visualizations only support a specific data type and/or attack (Best et al., 2014). In order to mitigate some of these adoption issues, the proposed service is designed to be flexible and work with any existing network flows that can also include extra crafted features.

Typically, obtaining a better understanding of the network can help ensure understanding of the overall trends of the network and obtain better situational awareness. With this better situational awareness,s the false positives can be more easily spotted. This can be done using exploratory data analysis that can observe the statistical properties of the data to find overall trends to help spot anomalies (Collins & Collins, 2014). Visualization tools can be a great way to perform exploratory data analysis. They can help the analysis by offering increased data density and allowing analysts to get high level results for decision making by presenting large amounts of data in summarized form (Zhang et al., 2017). Visualizations can be a great way of representing a large amount of data using less space than text-based analysis. They also allow for more open-ended investigations by using techniques such as exploratory data analysis that enables users to examine the data and find trends they do not yet know to look for. Exploring the data visually has been shown to increase the effectiveness of network security data analysis (Goodall, 2009).

Most of an analyst's time is spent responding to incidents which usually have priority over exploring data for new attacks (Franklin et al., 2017). When analysts use exploratory data analysis, they can both respond to current alerts as well as answer hypothesis about the broader set at the same time. Linked visualization dashboards have been used for network security analysis that encourage users to discover patterns in the data (Zhang et al., 2014). Approaches such as this focus on offering useful visuals to help analysts gain situational awareness but do not offer any backend support for detection of trends. They provide interactive visualizations with the expectation that users perform the exploratory data analysis on their own. Other types of cybersecurity visualizations are instead centered on alert visualization and act as an interface to interpret classifier alerts. The proposed service in this work is similar to both in that it combines both approaches to build a tool allowing for alert visualization in combination with exploratory data analysis. The main distinction with the approach proposed in this work is the focus on comparing multiple machine learning explanations rather than a single alerting system to discover relationships between detection methods.

ARCHITECTURE OVERVIEW

To ensure that an IDS system is helpful, the analysts need to be confident in the results of the classifier. Since this confidence is hard to obtain when using advanced machine learning algorithms, an analyst will simply see a record sent in as input and an anomaly score or class assignment as an output. The internal operations and decision criteria responsible for the scoring are usually unknown to the analyst. The proposed service architecture is organized in a way that clearly describes how outlier scores are built.

Guidance, in the context of visual analytics can be defined as ``a computer-assisted process that aims to actively resolve a knowledge gap encountered by analysts during an interactive visual analytics session.'' (Ceneda et al., 2017). The knowledge gaps the authors aim to solve is the interpretation of outlier scores. The dashboard is built around narrowing this gap by providing visualizations to support the analyst in interpreting the results of the service-oriented architecture.

Here the authors present an example implementation of the framework as a real time interactive dashboard with the benefit of being able to perform what-if analysis scenarios in real time. At any point, the analyst can filter the dataset and gather statistics about the different contexts in real time rather than constantly querying a database, allowing for quick interpretation of outlier scores. Using interactive visualizations, the analyst can quickly transition between the output of each record and determine under which contexts the core record can be considered an anomaly.

The system is a combination of Python and JavaScript that is connected using Flask. The web dashboard visualizations are done using D3.js and the website formatting uses the web templating library bootstrap so that the site should be responsive and work for both desktop and mobile. The server uses web sockets to provide real-time two-way communication between the python backend and the visualization dashboard. An overview of the architecture can be seen in Figure 1. This service architecture design was chosen to easily facilitate communications between the end client and the service provider. The analytics can be done on a central server that can deliver the results to a web browser on any device. The web browser enables an interactive visualization dashboard to present the results of the service to the user. The service interface translates user interactions into requests to the backend server that performs the network flow context services.

The architecture enables the delivery of the service as black box where the user does not need to know the underlying details of the machine learning algorithms applied as they are abstracted into the service. In this way the service can be model-agnostic where different machine learning algorithms can be applied. This allows for greater flexibility when using the tool as the user can swap algorithms as the need arises.

SERVICE DESCRIPTION

As seen in Figure 2 there are four main steps to the application flow. The first step involves the analyst selecting which records on which to perform the analysis. The next step generates the contexts for

Figure 1. Architecture diagram

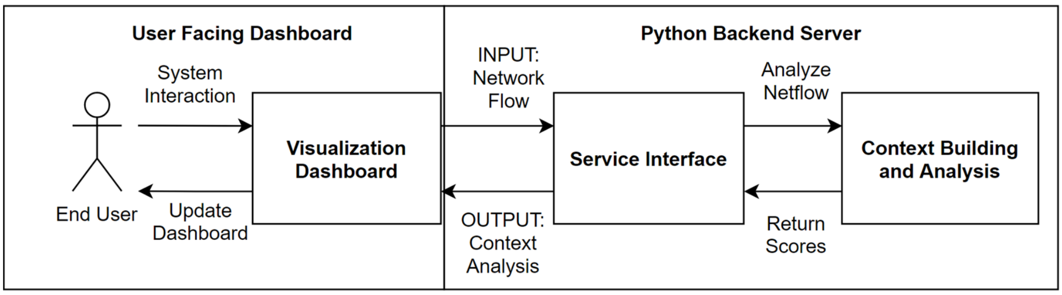
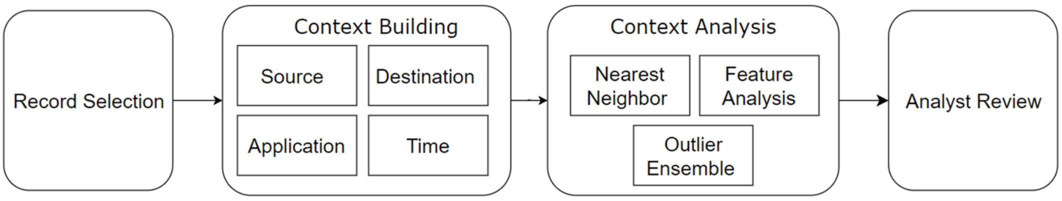


Figure 2. Service application flow



the record according to several features. Then for each of the contexts several evaluation steps are performed including nearest neighbor, feature analysis and a set of outlier algorithms. Lastly, the results are returned to the analyst for review. The following sections explain each of these steps in detail.

Record Selection

The first step is for the analyst to select network flow records on which analysis needs to be performed. This service architecture is designed to be used after an existing NIDS technique has been applied to the network data. The service is performed on records on an individual scale meaning that each individual network flow selected has its own instance of the service performed. This means that only a small subset of records should be selected to be used for the service as choosing every record would be very costly both in terms of time and processing requirements.

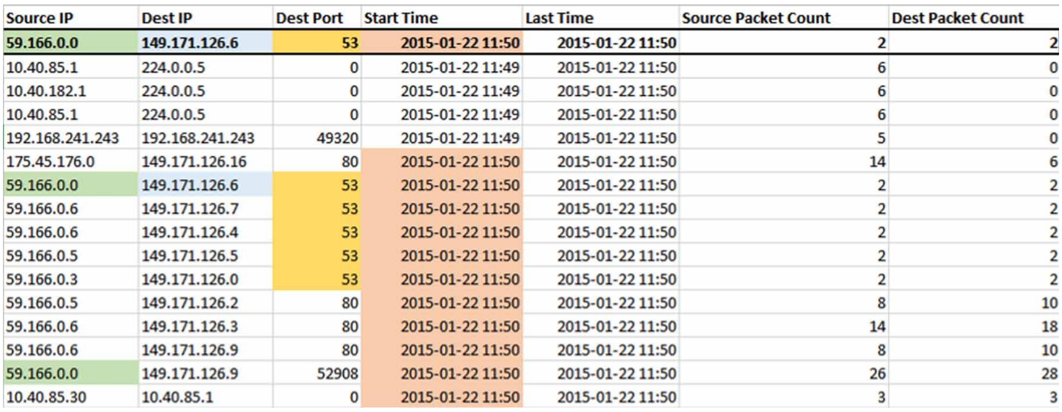
An analyst would perform the analysis on records of the highest priority as defined by them. One suggested way of prioritizing is to use any existing outlier scores for the records as a starting point. Since the framework only uses the features inherent in the network data to generate the analysis, the outlier scores can come from any type of detection system. When outlier scores are provided alongside the records, the system can use these to suggest areas of interest to investigate. Methods include ranking the records with the highest outlier scores or if multiple scores are presented, it can suggest records with the highest spread between outlier scores. Once one or more records have been selected, the architecture executes a service instance for each record.

Context Building

For each selected record which will be referred to as the core record, a group of contexts is generated relating to this record. These contexts are formed by taking all records in the entire dataset that match certain features of the core record. These features are the source host, destination host, application and time. The first two contexts match the source and destination address of the network flows. Included in each of these contexts is all the records matching the source IP address and destination IP address respectively. These contexts help identify any irregularities from within the scope of each endpoint of the network connection. The application context is built by matching the destination port of the core record with the rest of the dataset. This means that this context contains all records with the same destination port as the core record. The time context is built using all timestamps from the same time interval as the core record; in this case, the default setting chosen is 1 minute. Lastly, there is by default a global context that includes all records in the dataset and is used to provide a broad overview of how the record matches the properties of the entire network.

An example of context building can be seen in Figure 3. The header row of the table displays the feature names and the second row has been outlined and contains the core record that has been selected to perform the service on. The core record has a source IP address of 59.166.0.0. This means that when building the source address context all rows matching this address are included. All matching records are depicted as being highlighted in the first column. The same operation is performed for the destination address of 149.171.126.6 with matching records highlighted in the second column. For the application context, the core record has a destination port address of 53 (used for translating

Figure 3. Context building example



Source IP	Dest IP	Dest Port	Start Time	Last Time	Source Packet Count	Dest Packet Count
59.166.0.0	**149.171.126.6**	**53**	**2015-01-22 11:50**	**2015-01-22 11:50**	2	2
10.40.85.1	224.0.0.5	0	2015-01-22 11:49	2015-01-22 11:50	6	0
10.40.182.1	224.0.0.5	0	2015-01-22 11:49	2015-01-22 11:50	6	0
10.40.85.1	224.0.0.5	0	2015-01-22 11:49	2015-01-22 11:50	6	0
192.168.241.243	192.168.241.243	49320	2015-01-22 11:49	2015-01-22 11:50	5	0
175.45.176.0	149.171.126.16	80	2015-01-22 11:50	2015-01-22 11:50	14	6
59.166.0.0	149.171.126.6	53	2015-01-22 11:50	2015-01-22 11:50	2	2
59.166.0.6	149.171.126.7	53	2015-01-22 11:50	2015-01-22 11:50	2	2
59.166.0.6	149.171.126.4	53	2015-01-22 11:50	2015-01-22 11:50	2	2
59.166.0.5	149.171.126.5	53	2015-01-22 11:50	2015-01-22 11:50	2	2
59.166.0.3	149.171.126.0	53	2015-01-22 11:50	2015-01-22 11:50	2	2
59.166.0.5	149.171.126.2	80	2015-01-22 11:50	2015-01-22 11:50	8	10
59.166.0.6	149.171.126.3	80	2015-01-22 11:50	2015-01-22 11:50	14	18
59.166.0.6	149.171.126.9	80	2015-01-22 11:50	2015-01-22 11:50	8	10
59.166.0.0	149.171.126.9	52908	2015-01-22 11:50	2015-01-22 11:50	26	28
10.40.85.30	10.40.85.1	0	2015-01-22 11:50	2015-01-22 11:50	3	3

domain names to IP addresses). Other ports could indicate activity such as web traffic or file transfers; however, since the core record is using port 53, we are only extracting all traffic relating to this specific port. All matching records for the context are highlighted in the third column. Lastly, the start time occurs at 11:50am and the 1-minute interval scope is any record within the same minute as seen highlighted in the fourth column.

It can be seen that some records share many features with the core record; this means they are placed in many of the contexts and therefore will have a greater influence on the outlier ranking of the core record. Other records that do not have any matching features will not be part of any contexts except for the default global context meaning their influence on the core record will be much less significant.

Context Analysis

Once the contexts have been made, the next step is the evaluation phase. For each of the contexts, three operations are performed: the nearest neighbors to the core record are identified, the feature distributions are analyzed, and an ensemble of unsupervised machine learning algorithms are used on the context.

Nearest Neighbor

Finding the nearest neighbors for a record means that the service is comparing the similarity of the core record with every other record in the context. By determining the relationship between all records with the core record, the service can quickly find any records that are quite similar or even nearly identical to the core record. If such cases exist, the analyst can handle these instances at the same time as the core record, as the appropriate response would likely be the same with such similar records. With the calculation of the nearest neighbors, the analyst has the opportunity to address all these points as a group decision, saving time to respond to more unique events.

In instances where the number of features present in the dataset is high, the service will first perform dimension reduction prior to calculating distances between points. This is due to the curse of dimensionality where records in high dimensions are all far away from one another. The analysis as an output provides a ranked list of record indexes and their distance from the core record. The records are sorted by distance with a smaller distance indicating a closer relationship to the core record with a smallest possible value of 0. An example output can be seen below with a unique identification number of each record as the key and the distance to the core record as the value. The first entry with the index of "3203" is the closest record with a distance of 0.0392.

```
{
“3203”: 0.0392,
“5422”: 0.0491,
“0929”: 0.0735,
“2332”: 0.1236
}
```

Feature Distributions

The feature analysis operation involves measuring the distributions of the features for each context. The comparison is done using all the features of the dataset to get a general overview as to how the core record relates to the rest of the records in the context. Statistics including the minimum, average, maximum and standard deviation are done for each feature in the dataset. These statistics are then compared to the core record to identify any significant deviations from the overall average for the context. To reduce the number of features analyzed, techniques including recursive feature elimination and dimension reduction can be performed.

An example output for one feature of a context follows. The context and feature are identified as well as the minimum, average and maximum for the context as well as the value the core record has for this feature and lastly the number of standard deviations the value is from the mean. In this case, the number of bytes sent in the connection is being analyzed within the context of the source host.

```
{
“feature”: “sBytes”,
“context”: “source_ip”
“mean”: 21.0125,
“max”: 10200.0,
“min”: 1.0,
“record_value”: 92,
“record_std”: 0.3332,
}
```

Outlier Ensemble

The last analysis phase is to run the context through an ensemble of outlier detection algorithms. This is a collection of various unsupervised machine-learning algorithms that are being used to detect outliers within the dataset. Each algorithm outputs an outlier score for each record that indicates the degree to which each point is considered anomalous. The machine learning algorithms included in the ensemble are an isolation forest, Local Outlier Factor (LOF) and HDBSCAN.

The isolation forest works by selecting a feature at random and then splitting the dataset by that feature at a random value. These splits form a tree structure that is then traversed to find the depth that is required for a point to be separated from the rest of the dataset. The path length that is averaged over a forest of many trees is used to generate the outlier score. Anomalous points tend to be separated faster meaning that shorter paths generate a higher outlier score.

Local Outlier Factor is an algorithm that detects outliers by comparing each record to its neighbors and is focused on the concept of local density. The local density is measured as the average distance to the nearest neighbors of the point. With this local approach LOF can find outliers in local subsets of the data that may not be considered anomalous in another subset due to the relative densities of its neighbors in the area.

HDBSCAN is a hierarchical clustering algorithm based off DBSCAN. An advantage of this algorithm to simpler methods such as K-means is the ability to form arbitrary cluster shapes as well as deal with noise by having some points not placed in any cluster. This algorithm supports the GLOSH outlier detection algorithm that operates in a similar manner to LOF and is used to generate outlier scores from the resulting clustering.

For each context, each of these outlier detection algorithms is ran. The context and algorithm tested are stored as well as the average outlier score for the context, the score of the core record and the difference between these scores. Below is an example output for one model in one of the contexts. This is one of the results for the source IP address context, in this instance the isolation forest algorithm. The outlier scores have a range between 0 representing an entirely normal record and 1 indicating a completely anomalous record. Here the average score for all records in the context is a score of 0.1352, the record has a score of 0.0101 with the record have a score 0.125 below the average indicating it is less anomalous than the average record.

```
{
“context”: “source_ip”,
“model”: “isolation_forest”,
“average_score”: 0.1352,
“record_score”: 0.0101,
“score_difference”: -0.125
}
```

Analyst Review

Every context has the following outputs, one file each for feature analysis, outlier ensemble and list of nearest neighbors. Additionally, a summary log is provided that aggregate the data across all contexts into two summary statistics. The first is an uncertainty metric that measures the extent to which the classifiers are consistent in the outlier scores assigned to the core record. This is calculated as the gap between the highest and lowest assigned outlier scores. As an example of the most anomalous model score was from the isolation forest with a score of 0.90 and the lowest was from HDBSCAN with 0.05 the outlier gap is 0.85. The maximum gap would be a score of 1.0, in this instance a score of 0.85 indicates a large amount of uncertainty within the ensemble.

Outlier Score Gap = (Largest outlier score) – (Smallest Outlier Score)
= (Isolation Forest (0.90)) – ((HDBSCAN 0.05))
= 0.85

The second measure is a summary score that can be used to give a high-level overview of how anomalous the record is across all contexts and outlier algorithms. This score is calculated as the average amount of standard deviation the core record has for each feature multiplied by the average outlier score assigned to the core record. The absolute values are used for the standard deviations as the system does not differentiate between being above or below the average. As an example, a record with standard deviations including 0.9, 1.3, 0.5 and 0.8 would average to 0.875, the outlier ensemble with scores of 0.90, 0.50 and 0.05 would average to 0.483. When multiplied the resulting score is 0.422.

Summary Score = (Average Standard Deviation) * (Average Outlier Score)
= (average (0.90 + … + 0.8)) * (average (0.90 + 0.50 + 0.05))
= (0.875) * (0.483)
= (0.422)

USE CASES

Analyzing Outlier Score Distributions

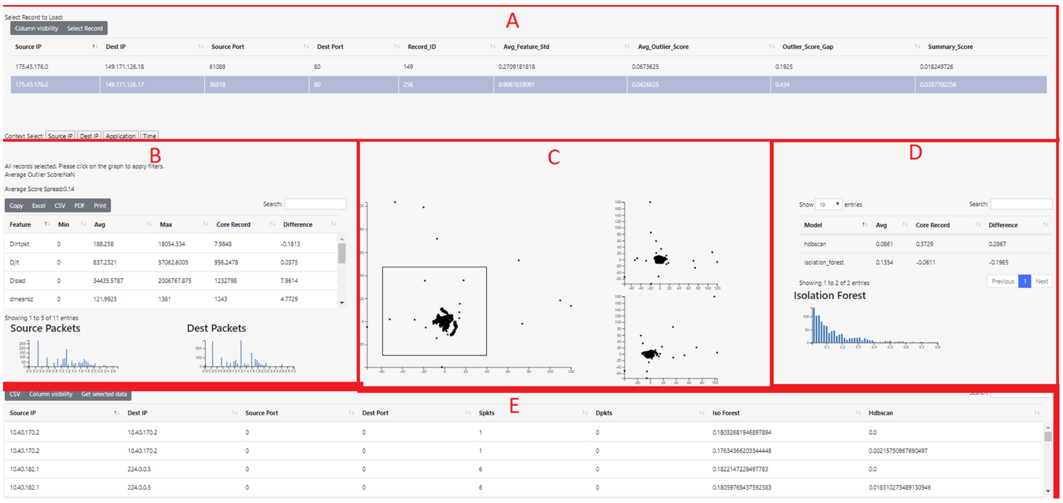
As seen in Figure 4, the dashboard is divided into five main components: the service results selection (3A), the feature analysis and outlier ensemble data tables (3B, 3D)), and nearest neighbor scatterplots (3C), and the nearest neighbors data table (3E). All of these groups are connected together in a single linked visualization dashboard. Every view (graph) used in the dashboard supports brushing and linking of the views on the same set of data. A change to one view will change the selection in the other linked views so that when one graph is filtered the other graphs are updated accordingly.

Service Results Selection

This area is where the analyst will choose from all the records that have been run through the service. Once a record has been chosen the analyst then chooses which context to examine and can compare between the various contexts. As seen in Figure 5, the data table lists the main features of the network flow as well as the average feature standard deviation, the average of the assigned outlier scores as well as the outlier gap and summary score calculations.

An analyst may wish to examine instances where there is a large gap between the outlier scores assigned by the components of the ensemble. This gap between the most confident and least confident anomaly detectors can indicate interesting scenarios where an ensemble was unsure of the extent to which the record was an outlier and could be a good candidate for enhanced review by the analyst.

Figure 4. Dashboard overview



Select Record to Load:

Source IP	Dest IP	Source Port	Dest Port	Record_ID	Avg_Feature_Std	Avg_Outlier_Score	Outlier_Score_Gap	Summary_Score
175.45.176.0	149.171.126.18	61089	80	149	0.2709181818	0.0673625	0.1925	0.018249726
175.45.176.0	149.171.126.17	36918	80	256	0.9087659091	0.0426625	0.434	0.0387702256

Context Select: Source IP | Dest IP | Application | Time

All records selected. Please click on the graph to apply filters.
Average Outlier Score:NaN

Average Score Spread:0.14

Feature	Min	Avg	Max	Core Record	Difference
Dintpkt	0	188.258	18054.334	7.9848	-0.1813
Djit	0	837.2321	57062.6005	956.2478	0.0375
Dload	0	34435.5787	2006767.875	1232798	7.9614
dmeansz	0	121.9925	1381	1243	4.7729

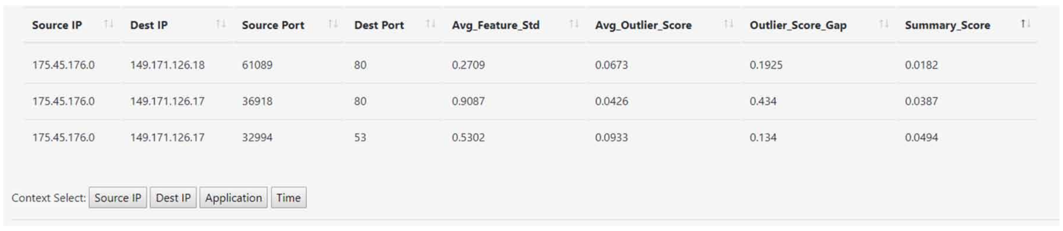
Showing 1 to 5 of 11 entries

Model	Avg	Core Record	Difference
hdbscan	0.0861	0.3729	0.2867
isolation_forest	0.1354	-0.0611	-0.1965

Showing 1 to 2 of 2 entries

Source IP	Dest IP	Source Port	Dest Port	Spkts	Dpkts	Iso Forest	Hdbscan
10.40.170.2	10.40.170.2	0	0	1	0	0.18032681946897894	0.0
10.40.170.2	10.40.170.2	0	0	1	0	0.17634366203344448	0.00215750967690497
10.40.182.1	224.0.0.5	0	0	6	0	0.1822147228497783	0.0
10.40.182.1	224.0.0.5	0	0	6	0	0.18059768437592383	0.018310273489130946

Figure 5. Service results table



Source IP	Dest IP	Source Port	Dest Port	Avg_Feature_Std	Avg_Outlier_Score	Outlier_Score_Gap	Summary_Score
175.45.176.0	149.171.126.18	61089	80	0.2709	0.0673	0.1925	0.0182
175.45.176.0	149.171.126.17	36918	80	0.9087	0.0426	0.434	0.0387
175.45.176.0	149.171.126.17	32994	53	0.5302	0.0933	0.134	0.0494

Context Select: Source IP | Dest IP | Application | Time

To examine these records more closely one can use the outlier gap score and the summary score computed by the service.

Feature Analysis

As seen in Figure 6, this section shows the output of the feature analysis and outlier ensembles for the chosen context. On the left side is a data table with a row for each feature analyzed. At the bottom of this section is a histogram that is used to display in detail the distribution of specific features. By default, the feature in which the core record is most considered anomalous is chosen but any feature can be swapped by selecting it in the data table. The right side of Figure 4 displays the output of outlier scores from each model in the outlier ensemble used with the service. Just as with the features, every model can be represented as a histogram that shows the distribution of outlier scores with anomaly rating on the x-axis from least anomalous (0) to most (1) and the y-axis plots the amount of records matching each bin. The histograms have adjustable bin sizes to identify different levels of distribution in the features.

Using the histograms, the analyst can filter the context to take all records matching a specific subset of a feature. This can be done by brushing over specific ranges in the feature histograms. For example, the analyst may select the records with a high amount of source packet counts. Taking slices of the histogram selects the threshold value, where scores outside this range are filtered out. As an example, the analyst may wish to identify all the nearest neighbors that are also marked as likely to be outliers by taking the values with high outlier scores and seeing how the feature distribution histograms change.

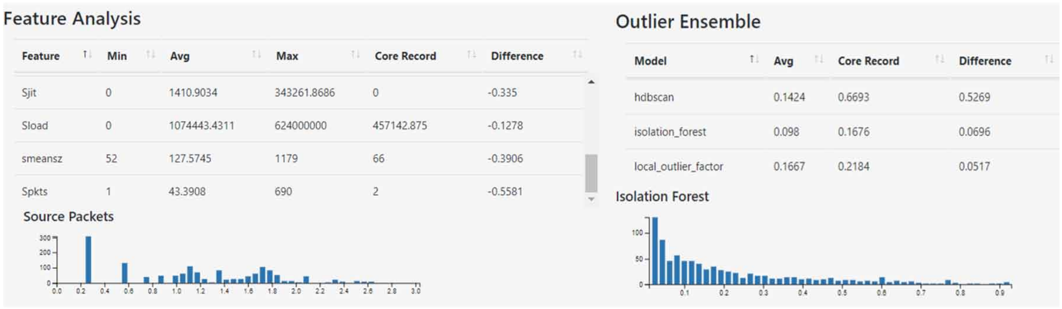
Visualizing Network Flow Neighborhoods

Nearest Neighbor Data Table

The nearest neighbor data table lists the details of every record part of the nearest neighbors. Every column can be sorted by selecting the column header, with multiple column sorting also supported. Each column can also be filtered using regular expressions for more precise filtering requirements.

In addition to the base contexts generated from the service, the analyst can build their own custom contexts using the linked visualizations in the interactive dashboard. Using regular expressions in the nearest neighbor data table, the analyst may wish to sort by a specific host or application during a certain time period, thereby creating their own custom sub context within the currently selected context. For example, while browsing the source address context they may apply a filter for a specific port to further narrow in on interesting traffic and are now viewing traffic from the same application on the same source device. As seen in Figure 7 the second column has been sorted using the IP address subnet 149.171.126 meaning any address starting with these digits is displayed. Additionally, the data is also filtered according to the destination port of 80 in the fourth column so only HTTP traffic has been selected.

Figure 6. Feature analysis and outlier ensemble result tables



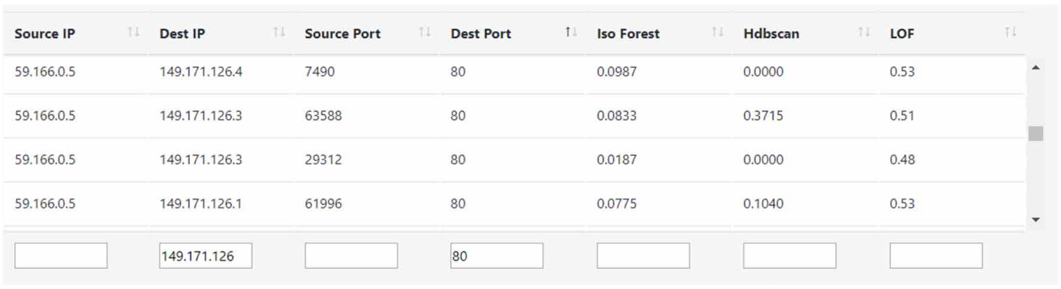
Feature Analysis

Feature	Min	Avg	Max	Core Record	Difference
Sjit	0	1410.9034	343261.8686	0	-0.335
Sload	0	1074443.4311	624000000	457142.875	-0.1278
smeansz	52	127.5745	1179	66	-0.3906
Spkts	1	43.3908	690	2	-0.5581

Outlier Ensemble

Model	Avg	Core Record	Difference
hdbscan	0.1424	0.6693	0.5269
isolation_forest	0.098	0.1676	0.0696
local_outlier_factor	0.1667	0.2184	0.0517

Figure 7. Nearest neighbor data table



Source IP	Dest IP	Source Port	Dest Port	Iso Forest	Hdbscan	LOF
59.166.0.5	149.171.126.4	7490	80	0.0987	0.0000	0.53
59.166.0.5	149.171.126.3	63588	80	0.0833	0.3715	0.51
59.166.0.5	149.171.126.3	29312	80	0.0187	0.0000	0.48
59.166.0.5	149.171.126.1	61996	80	0.0775	0.1040	0.53
	149.171.126		80			

Dimension Reduction Scatterplots

Figure 8 contains scatterplots that displays all of the nearest neighbors of the core record dataset. With this dashboard the authors use dimension reduction as a tool used to represent high dimensional datasets in a format that can be displayed in two dimensional scatterplots. Using the t-SNE dimension reduction algorithm, the three scatterplots depict a two-dimensional representation of the original full dimensional data. The scatterplots show the data reduced to three dimensions with each graph plotting the different combinations of the three dimensions. The scales are arbitrary with the importance being the relative distances between points. The proximity of points is a correlation to their similarity in the full dimension set meaning that points close to each other have more in common than points that are a further distance apart.

The scatterplot supports brushing where the analyst can drag a box over the scatterplot to select points. The dataset will be filtered down to the points within the drawn box. Records can be examined in groups or as individual records. Grouping data points allows one to build contexts that can be compared to one another to identify subsets of the data that exhibit similar patterns. To select the most similar records one selects a range surrounding the core record as a drag able box onto the scatterplot. Now only those records within the bounds visually set by the drawn box are part of the selection. All other records have been filtered out of the working set. By brushing over different clusters, the similarities in their features are visible in the histograms found in the other linked views. Finding the nearest neighbors of the core record allows the analyst to quickly resolve this record as well as quickly assess if any similar instances exist.

The core record is the point that has been circled. Using these scatterplots the analyst can determine how similar the record is to the rest of the context. In Figure 8 the record is separated from the rest of the group in all three dimensions. This means that the record is overall different than the majority of the records. As these scatterplots are using reduced dimensions this is a high-level overview of the similarity, providing the analyst a big picture view of the relationship between the core record and the rest of the network.

Anomalous Feature Scatterplots

In contrast to the dimension's reduction scatterplots, the scatterplots in Figure 9 display specific features on the axis. The features can be chosen by the analyst or automatically filled by selecting the features that the core record has the farthest distance from the mean of the feature. In this example the scatterplots display the three combinations of source packet count, destination packet count and source byte count. The core record is the point that has been circled. Using these scatterplots, the analyst can determine how similar the record is to the rest of the context according to these features. This allows an analyst to understand what makes this record anomalous by visually seeing how separated the record is from the rest of the records.

Figure 8. Dimension reduction scatterplots

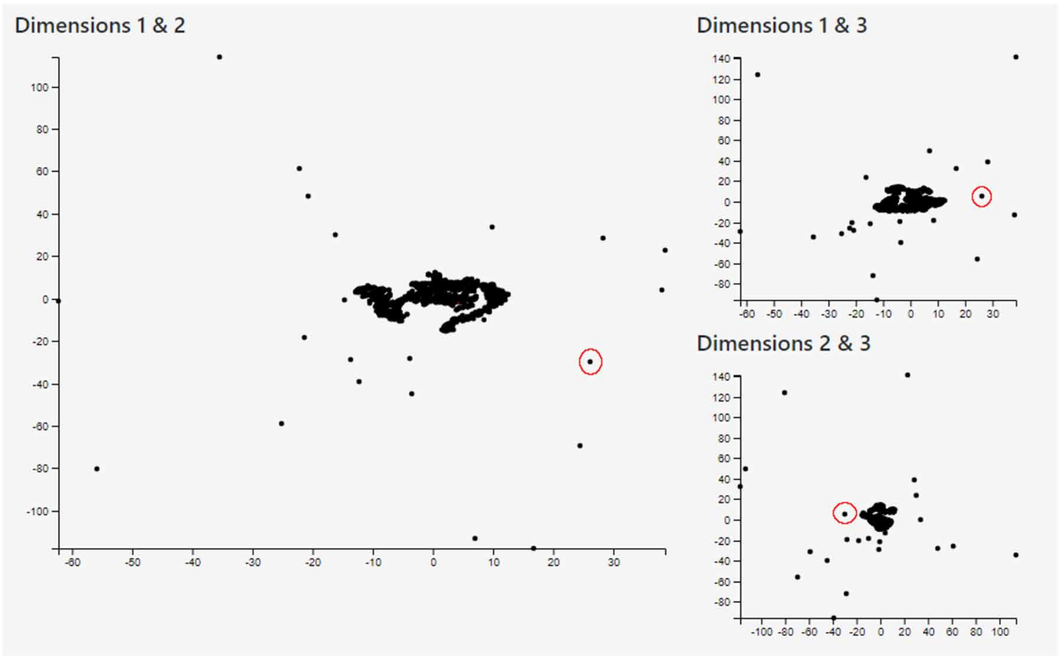
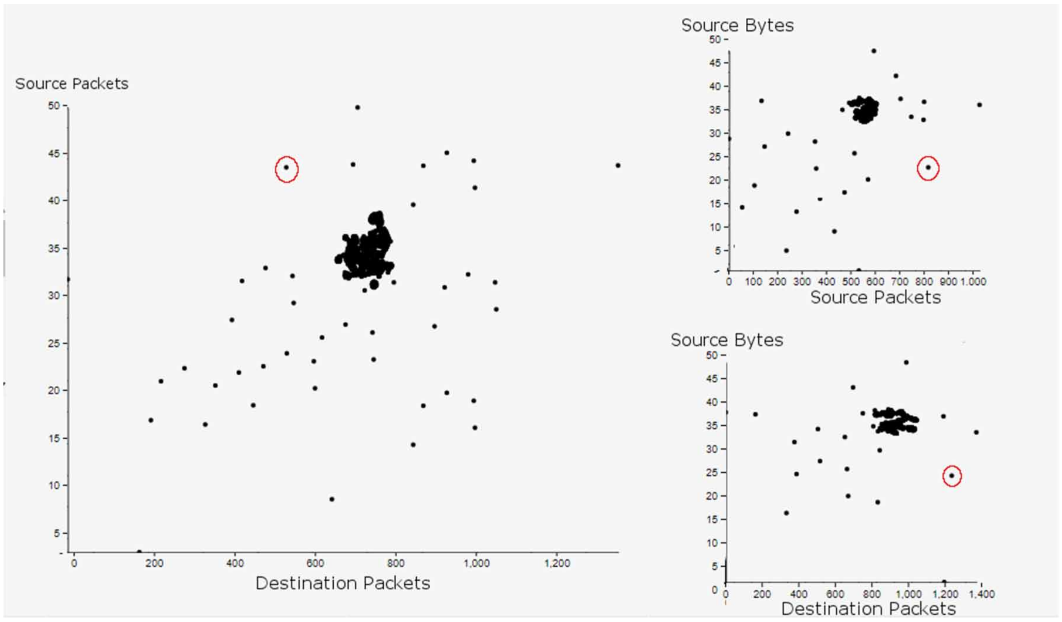


Figure 9. Feature scatterplots



EVALUATION

Network Flow Data Set

In this work the authors are using the UNSW-NB15 Dataset (Moustafa & Slay, 2015; Moustafa & Slay, 2016) to show how the system would work. This is a network flow dataset from The University of New South Wales with the cyber security research group at the Australian Centre for Cyber Security (ACCS). The traffic was generated from a network simulator combined with a synthetic attack generator. The records were formed from raw packet captures (pcaps) which then was sorted into netflows. It has 49 features per record including the standard netflow entries (matching IP, ports and protocol between devices) and extra crafted features from Bro-IDS and custom C# scripts. The attack classes are analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode and worms. More details about data collection, features and classes can be found in (Moustafa & Slay, 2015; Moustafa & Slay, 2016).

Performance Metrics

Classification results can be placed into four categories based on whether the predicted class matches the true class. True positives and negatives are the ideal results while false positives and negatives represent false alarms and missed attacks.

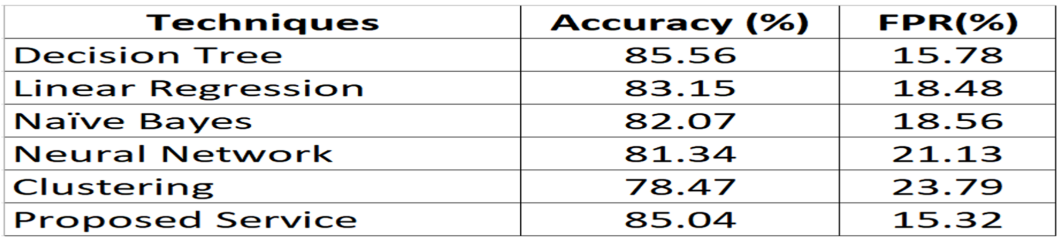
- True Positive (TP) - Both the predicted class by our classifier and the true label are positive. In our context the positive class is attack traffic, so this represents a successfully identified attack.
- True Negative (TN) - Both the predicted class by our classifier and the true label are negative. In our context the negative class is regular traffic, so this represents a traffic that has been properly interpreted as normal.
- False Positive (FP) - The classifier has incorrectly assigned the positive (attack) class to a record that is actually part of the negative (normal) class. This represents false alarms that have incorrectly triggered as an attack.
- False Negative (FN) - The classifier has incorrectly assigned the negative (normal) class to a record that is actually part of the positive (attack) class. This represents attacks that have evaded detection.

The proposed service is compared against evaluation done by the authors of the NB15 dataset (Moustafa & Slay, 2016). The metrics used for evaluation are accuracy and false positive rate. Accuracy is the simplest metric and is the fraction of correct results over all records. This is true positives and true negatives divided by all of the records. False Positive Rate (FPR) is the percentage of normal traffic that was classified as an attack. Also known as the inverse to "specificity". This is calculated as the false positives divided by the false positives and true negatives.

$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$
$\text{FPR}= (FP)/(FP+TN)$

As seen in Figure 10, the results of the service proposed in this work is compared with existing algorithms. The first five rows show the test run by the dataset authors on the highest rated classifier for each machine learning type tested dataset (Moustafa & Slay, 2016). The last row compares our service on the same dataset. The proposed service is within the performance range for the accuracy and false positive rate of existing approaches. This demonstrates that our service can provide extra context to network flows without a trade off in detection accuracy or an increase in false positives.

Figure 10. Performance comparison



Techniques	Accuracy (%)	FPR(%)
Decision Tree	85.56	15.78
Linear Regression	83.15	18.48
Naïve Bayes	82.07	18.56
Neural Network	81.34	21.13
Clustering	78.47	23.79
Proposed Service	85.04	15.32

Limitations and Future Work

This service-oriented architecture is intended to be used as an additional component of an existing intrusion detection system and not as a complete standalone solution. The use of this service is very processing intensive and therefore is not to be used on every network record. It is important for analysts to have clearly defined priorities that will be used as decision criteria for which data is to be processed using the architecture. Responsiveness of the dashboard slows considerably when using a large amount of records in the scatterplots. This limits the scalability of the scatterplot to scale of thousands however this design limitation is mitigated by focusing the use of the scatterplot only on the nearest neighbors of the selected record.

In future iterations of this service architecture the authors plan to further explore the guidance provided by the system. Since the experience of each analyst can vary greatly as well as the overall context and objective at the time, additional future work includes a system that will offer different levels of guidance. For example, a ML novice may be given a more automated result with the system determining the best results for their needs. Those will greater experience may opt to have the system simply present some of the best options and allow the analyst to manually inspect each approach on their own to come to their own conclusion as to the best approach. Future work also includes doing performance evaluations on the various components to determine which contexts, ML algorithms, ensemble normalization and combination functions and features provide the best detection rates as well as analyst comprehension.

CONCLUSION

This paper proposes a service architecture that helps to provide a way to better interpret outlier scores for the purposes of network intrusion detection. This is done by taking network flows and building contexts around them by matching specific features of the dataset with the feature values of the individual network flow record. These data subsets are then run through feature distribution analysis, a nearest neighbor algorithm and an ensemble of unsupervised outlier detection algorithms to provide a deep explanation as to how specific records are considered anomalous.

The authors then provide an example implementation of this service in the form of a linked visualization dashboard using the UNSW-NB15 network flow dataset. The dashboard combines the results in data tables, histograms and scatterplots to visually represent the results to the analyst. The results of the service can be sorted using the calculated summary score and outlier gap metrics. This provides the analyst a quick method of prioritizing the service analysis. With a service output selected the analyst can then compare the feature analysis and outlier algorithms from the various contexts and display these in histograms to analyze the distributions. A data table provides information on all of the nearest neighbors allowing the analyst to address these similar instances as well.

Scatterplots are used to visually depict the separation of the core record from the rest of the data in the contexts. Dimension reduction scatterplots display the data as represented in three compressed dimensions. This provides an overall approximation of record similarity. For more detailed differences the scatterplots display specific features that are anomalous for the core record. These graphs provide visual insight as to how the core record can be considered an outlier. Using this service architecture an analyst can assess network flows using a wide variety of machine learning approaches and using the dashboard application, can combine this with visualizations to better understand outlier scores.

ACKNOWLEDGMENT

This research was supported by the Communications Security Establishment, the government of Canada's national cryptologic agency.

REFERENCES

Aggarwal, C., & Sathe, S. (2017). *Outlier ensembles. ACM SIGKDD Explorations Newsletter*, 14. doi:10.1007/978-3-319-54765-7

Aggarwal, C. C. (2013). *Outlier Analysis*. New York, NY: Springer New York. doi:10.1007/978-1-4614-6396-2

Angelini, M., Aniello, L., Lenti, S., Santucci, G., & Ucci, D. (2017). The goods, the bads and the uglies: Supporting decisions in malware detection through visual analytics. *Proceedings of the 2017 IEEE Symposium on Visualization for Cyber Security (VizSec)* (pp. 1-8). IEEE. doi:10.1109/VIZSEC.2017.8062199

Beaugnon, A., Chifflier, P., & Bach, F. (2017, September). Ilab: An interactive labelling strategy for intrusion detection. *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses* (pp. 120-140). Springer. doi:10.1007/978-3-642-33338-5

Best, D. M., Endert, A., & Kidwell, D. (2014). 7 Key Challenges for Visualization in Cyber Network Defense. *Proceedings of the Eleventh Workshop on Visualization for Cyber Security* (pp. 33–40). Academic Press. doi:10.1145/2671491.2671497

Buczak, A., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys and Tutorials*, *18*(2), 1153–1176. doi:10.1109/COMST.2015.2494502

Ceneda, D., Gschwandtner, T., May, T., Miksch, S., Schulz, H. J., Streit, M., & Tominski, C. (2017). Characterizing Guidance in Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 111–120. doi:10.1109/TVCG.2016.2598468 PMID:27514054

Cisco Systems. (2018). *Annual Cybersecurity Report.*

Collins, M., & Collins, M. S. (2014). *Network security through data analysis: building situational awareness*. O'Reilly Media, Inc.

Endert, A., Ribarsky, W., Turkay, C., Wong, B. L. W., Nabney, I., Blanco, I. D., & Rossi, F. (2017). The State of the Art in Integrating Machine Learning into Visual Analytics. *Computer Graphics Forum*, *36*(8), 458–486. doi:10.1111/cgf.13092

Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2019). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447–489. doi:10.1007/s11235-018-0475-8

Fink, G. A., North, C. L., Endert, A., & Rose, S. (2009). Visualizing cyber security: Usable workspaces. *Proceedings of the 2009 6th international workshop on visualization for cyber security* (pp. 45-56). IEEE. doi:10.1109/VIZSEC.2009.5375542

Franklin, L., Pirrung, M., Blaha, L., Dowling, M., & Feng, M. (2017, October). Toward a visualization-supported workflow for cyber alert management using threat models and human-centered design. *Proceedings of the 2017 IEEE Symposium on Visualization for Cyber Security (VizSec)* (pp. 1-8). IEEE.

Goodall, J. R. (2009). Visualization is better! a comparative evaluation. *Proceedings of the 2009 6th International Workshop on Visualization for Cyber Security* (pp. 57-68). IEEE.

Habeeb, R. A. A., Nasaruddina, F., Ganib, A., Hashemb, I. A. T., Ahmed, E., & Imran, M. (2019). Real-time big data processing for anomaly detection: A Survey. *International Journal of Information Management*, *45*, 289–307. doi:10.1016/j.ijinfomgt.2018.08.006

Kang, D. (2007). Categorizing post-deployment IT changes: An empirical investigation. *Journal of Database Management*, *18*(2), 1–24. doi:10.4018/jdm.2007040101

Kaul, M., Storey, V. C., & Woo, C. (2017). A Framework for managing complexity in information systems. *Journal of Database Management*, *28*(1), 31–42. doi:10.4018/JDM.2017010103

Keim, D., Andrienko, G., Fekete, J. D., Görg, C., Kohlhammer, J., & Melançon, G. (2008). Visual analytics: Definition, process, and challenges. In *Information visualization* (pp. 154–175). Berlin: Springer. doi:10.1007/978-3-540-70956-5_7

Keim, D. A., Mansmann, F., & Thomas, J. (2010). Visual analytics: How much visualization and how much analytics? *ACM SIGKDD Explorations Newsletter*, *11*(2), 5–8. doi:10.1145/1809400.1809403

Keller, F., Muller, E., & Bohm, K. (2012). HiCS: High contrast subspaces for density-based outlier ranking. *Proceedings of the 2012 IEEE 28th international conference on data engineering* (pp. 1037-1048). IEEE. doi:10.1109/ICDE.2012.88

Khanuja, H. K., & Adane, D. (2018). Monitor and Detect Suspicious Transactions With Database Forensic Analysis. *Journal of Database Management*, *29*(4), 28–50. doi:10.4018/JDM.2018100102

Krause, J., Perer, A., & Bertini, E. (2016). Using Visual Analytics to Interpret Predictive Machine Learning Models. *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning (WHI)* (pp. 106-110). Academic Press.

Kriegel, H.-P., Kroeger, P., Schubert, E., & Zimek, A. (2011). Interpreting and Unifying Outlier Scores. *Proceedings of the 2011 SIAM International Conference on Data Mining* (pp. 13–24). Academic Press.

Kulesza, T., Burnett, M., Wong, W. K., & Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. *Proceedings of the 20th international conference on intelligent user interfaces* (pp. 126-137). ACM. doi:10.1145/2678025.2701399

Micenková, B., McWilliams, B., & Assent, I. (2014). Learning Outlier Ensembles: The Best of Both Worlds – Supervised and Unsupervised. *Proc. of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM.

Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. *Proceedings of the conference on fairness, accountability, and transparency* (pp. 279-288). ACM. doi:10.1145/3287560.3287574

Moustafa, N., & Slay, J. (2015). *UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW- NB15 network data set)*. doi:10.1109/MilCIS.2015.7348942

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1–3), 18–31. doi:10.1080/19393555.2015.1125974

Nguyen, H. V., Ang, H. H., & Gopalkrishnan, V. (2010). Mining outliers with ensemble of heterogeneous detectors on random subspaces. *Proceedings of the International Conference on Database Systems for Advanced Applications* (pp. 368-383). Springer. doi:10.1007/978-3-642-12026-8_29

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. D. (2017). Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 61–70. doi:10.1109/TVCG.2016.2598828 PMID:27875134

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. doi:10.1145/2858036.2858529

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144). ACM. doi:10.1145/2939672.2939778

Sacha, D., Sedlmair, M., Zhang, L., Lee, J. A., Peltonen, J., Weiskopf, D., & Keim, D. A. et al. (2017). What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, *268*, 164–175. doi:10.1016/j.neucom.2017.01.105

Schneider, B., Jäckle, D., Stoffel, F., Diehl, A., Fuchs, J., & Keim, D. (2017). *Visual integration of data and model space in ensemble learning* (pp. 15–22). IEEE. doi:10.1109/VDS.2017.8573444

Schubert, E., Zimek, A., & Kriegel, H. P. (2014). Generalized outlier detection with flexible kernel density estimates. *Proceedings of the 2014 SIAM International Conference on Data Mining* (pp. 542-550). Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973440.63

Secureworks. (2018). *Incident Response Insights Report*.

Shiravi, H., Shiravi, A., & Ghorbani, A. (2012). A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics*, *18*(8), 1313–1329. doi:10.1109/TVCG.2011.144 PMID:21876227

Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (pp. 305-316). IEEE Press. doi:10.1109/SP.2010.25

Strobelt, H., Gehrmann, S., Pfister, H., & Rush, A. M. (2017). Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, *24*(1), 667–676. doi:10.1109/TVCG.2017.2744158 PMID:28866526

Veeramachaneni, K., Arnaldo, I., Cuesta-Infante, A., Korrapati, V., Bassias, C., & Li, K. (2016). AI 2: Training a Big Data Machine to Defend. *Proceedings of the International Conference in Big Data Security on Cloud*. Academic Press. doi:10.1109/BigDataSecurity-HPSC-IDS.2016.79

Wang, W., & Siau, K. (2019). Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity: A Review and Research Agenda. *Journal of Database Management*, *30*(1), 61–79. doi:10.4018/JDM.2019010104

Zhang, T., Liao, Q., & Lei, S. (2014). Bridging the Gap of Network Management and Anomaly Detection through Interactive Visualization. *Proceedings of the 2014 IEEE Pacific Visualization Symposium* (pp. 253–257). IEEE Press. doi:10.1109/PacificVis.2014.22

Zhang, T., Wang, X., Li, Z., Guo, F., Ma, Y., & Chen, W. (2017). A survey of network anomaly visualization. *Science China. Information Sciences*, *60*(12). doi:10.1007/s11432-016-0428-2

Zimek, A., Campello, R. J. G. B., & Sander, J. (2014). Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations Newsletter*, *15*(1), 11–22. doi:10.1145/2594473.2594476

Zimek, A., Gaudet, M., Campello, R. J., & Sander, J. (2013). Subsampling for efficient and effective unsupervised outlier detection ensembles. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 428-436). ACM. doi:10.1145/2487575.2487676

Brandon Laughlin received his Bachelors of Networking and Information Technology Security at the University of Ontario Institute of Technology in 2015 and a Master of Information Technology Security in 2016. He is now a full time Computer Science PhD student at the University of Ontario Institute of Technology specializing in information technology security. His research interests include network security, visualization, and machine learning.

Karthik Sankaranarayanan is Assistant Professor of Operations Management at the University of Ontario Institute of Technology, Canada. Prof. Sankaranarayanan received his Ph.D. in Management majoring in Operations Management from the University of Lugano, Switzerland in 2011 and was visiting researcher at the New England Complex Systems Institute, Cambridge, MA. His research encompasses the study of complex adaptive systems using agent-based modeling, experimental design and other computational tools. His primary research is in the area of behavioral operations management and currently explores the application of different methodological tools to study behavior in humanitarian operations. Prof. Sankaranarayanan also uses machine-learning algorithms to analyze human decision-making in order to build better decision-aid models.

Khalil El-Khatib is an associate professor at the University of Ontario Institute of Technology (UOIT). Prior to UOIT, he was an assistant professor at the University of Western Ontario until July 2006. In Feb. 2002, he worked of Research Officer in the Network Computing Group (lately renamed the Information Security Group) at the National Research Council of Canada for two years and continued to be affiliated with the group for another two years. His current research interests include: smart communities for smart cities, Big Data, and security analytics, Security and privacy issues in wireless sensor network, mobile wireless ad hoc networks, and vehicular ad hoc networks, smart grid security, cloud computing, biometrics, ubiquitous computing environments, and e-health.