A Generic Framework for Detecting Interpretable Real-Time Anomalies in Network Traffic Data

by

Nicholas Dowmon

B.S., Northeastern University 2015

Submitted to the System Design and Management Program
in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Author . . . .
System Design and Management Program
January 17, 2020

Certified by. . . . .
Abel Sanchez
Principal Investigator, Research Scientist, Geospatial Data Center
Thesis Supervisor

Accepted by . . . .
Joan S. Rubin
Executive Director, System Design and Management Program

A Generic Framework for Detecting Interpretable Real-Time Anomalies in Network Traffic Data

by

Nicholas Dowmon

Abstract

In this thesis, I present the Ensemble Outlier Detection System, a new framework for detecting anomalies in multidimensional network traffic data. The system meets six requirements which ensure that the design can meet the needs of the sponsor organization's cybersecurity teams both now and in the future. In particular, this system improves on many existing anomaly detection systems by maintaining scalability for extremely large computer networks, and resiliency to non-stationary data, re-establishing its own baselines as the network changes over time. I also present the Explorer tool, designed for cybersecurity analysts to interpret the cause of high anomaly scores on certain data points and to annotate each data point atomically. Scalability is ensured by treating each field in a data point as independent of one another, and preliminary results suggest that this treatment will not affect system performance, as many anomalous data points exhibit multiple anomalous fields at a time, increasing the outlier predictions for the data point using recursive aggregation.

Thesis Supervisor: Abel Sanchez
Title: Principal Investigator, Research Scientist, Geospatial Data Center

Acknowledgments

First, I would like to thank Dr. Abel Sanchez for his guidance on this research. You helped me answer many important questions while affording me the flexibility to take this project in the direction I saw fit.

Thank you to Joan Rubin, along with all of the faculty and staff at MIT SDM. My time at MIT has far exceeded my expectations; thank you all for making SDM such a special program here MIT.

Thank you also to my research sponsor institution, without whom this project could not have been completed. In particular, thank you Clark, Yosyp, and Eric for patiently answering my questions when I joined the project, and for all of your extra effort in making this project a success.

Thank you Rishi Shah and Robert Durfee for joining me as research assistants this year. Each of you taught me more than you know, and this project would not have been such a success without your hard work.

Thank you to my parents, Jim and Dina, for all of your support. I could never have made it here without you.

Most importantly, thank you Catherine. You have been my support system for my time at MIT and I never could have done any of it without you. And thanks in advance for your excellent proofreading skills.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of JSON Snippets

THIS PAGE INTENTIONALLY LEFT BLANK

Acronyms

APTs Advanced Persistent Threats. 9, 15

CDC Cyber Defense Center. 1–5, 15, 18, 63

CDF Cumulative Distribution Function. 24, 26, 40, 41, 44

DoS Denial of Service. 8, 12

DHCP Dynamic Host Configuration Protocol. 3

IDSs Intrusion Detection Systems. 15

IPFIX IP Flow Information Export. 10, 69

NIDSs Network Intrusion Detection Systems. 15

NIST National Institute of Standards and Technology. 2, 3

PPF Percent Point Function. 24

SIEM Security Information and Event Management. 4, 9, 11, 16

TCP Transmission Control Protocol. 11, 12, 16, 61

XSS Cross-Site Scripting. 8

VPN Virtual Private Network. 1, 61

WAPs Wireless Access Points. 1, 15

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

The goal of this research is to develop a framework for detecting anomalies in network traffic data on highly complex computer networks. In today's business environment, every company is a technology company, and billions of daily interactions between network-connected devices on an organization's network expose a highly complex attack surface for malicious parties. This broad attack surface is combined with increasingly sophisticated adversaries, who consistently pose new and previously unseen forms of attacks (zero-day attacks). This research aims to use recent advancements in technology, as well as leveraging contextual expert advice from cybersecurity analysts in order to implement a new defense against adversarial attacks on computer networks.

1.1 Problem Statement

The Massachusetts Institute of Technology (MIT) is currently collaborating with a Fortune 500 company to create new tools for their Cyber Defense Center (CDC) Tools team to combat modern threats to their computer network. Presently, the CDC monitors over 250,000 internal IP addresses on its network, encompassing unique devices ranging from employee client machines, networked servers, IoT-connected sensors, visitor Wireless Access Points (WAPs), Virtual Private Network (VPN) servers, and more. Overall, the CDC monitors over 2 billion daily network interactions, represent-

ing both internal traffic as well as all traffic crossing the network boundary.

In order to understand the larger context of cybersecurity risk, the CDC defines their processes in terms of the National Institute of Standards and Technology (NIST) Cybersecurity Framework Version 1.1, shown in Figure 1-1 [1]. The framework segments cybersecurity activities into five distinct categories: Identify, Protect, Detect, Respond, and Recover.
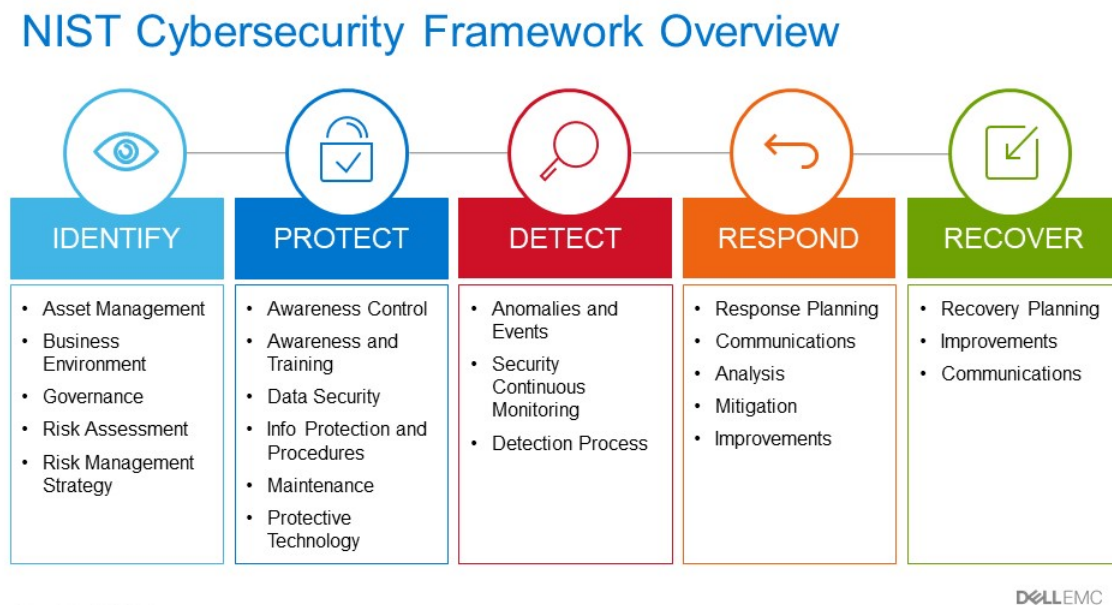
Figure 1-1: NIST Cybersecurity Framework Core, Version 1.1

Towards the goal of detecting zero-day attacks, the project deals primarily with the "Detect" function of the NIST Core - detecting anomalies and events. However, the functions "Identify", "Protect", and "Respond" are also relevant in detecting zero-day attacks, as they expose the network and organizational context which can improve the likelihood and impact of detecting previously unknown attacks.

The internal network is highly non-stationary, meaning the baseline behaviors change over time. With hundreds of thousands of personnel, devices are being added to and removed from the network daily. Moreover, with hundreds of global and distributed teams, the network structure is also non-stationary, with new subnetwork distinctions being made regularly on the network. These fluctuations constantly in-

troduce new behavior baselines to the overall network.

With such a dense and growing network, the CDC needs to effectively manage its resources to investigate only the highest-risk traffic interactions. Alerting the Cybersecurity Analyst for a potential threat in just 0.001% of the traffic on the network would result in over 20,000 threat investigations daily. Research shows that in the current practice, an average cybersecurity investigation takes 45 minutes to complete, suggesting that a fully allocated Cybersecurity Analyst can investigate approximately 10 threats in a given day. The time associated with investigations is primarily due to the context that Cybersecurity Analysts need to gather from a collection of isolated tools. For example, a Cybersecurity Analyst may first review Dynamic Host Configuration Protocol (DHCP) logs to understand which device is associated with a particular IP address, then identify the owner of that device, and finally look into an HR database to identify the owner's role. After gathering context from 30+ internal tools, the analyst decides whether the alert is benign or requires escalation. Many times, the investigation is inconclusive [2].

Because of the effort required with alert investigations, the CDC needs tools that raise alerts only for the most likely threats. However, the anomaly detection system should have enough precision to detect true attacks, and the suppression of false positives cannot also suppress the true positive anomalies in the system; suppressing true positive anomalies would reduce the anomaly detection system's recall (the percentage of actual attacks that it identifies), leaving a network vulnerable to unidentified attacks and the potential for devastating losses.

The NIST Cybersecurity Framework also provides guidance around how an organization views and responds to Cybersecurity Risk, called Implementation Tiers, which are described in Table 1.1 [3]. These tiers generally increase in proactivity, with Tier 1 representing ad-hoc and reactive teams with few organizational processes, and Tier 4 representing sophisticated cybersecurity practices which continuously improve by organizational learning and predictive indicators. In order to remain ahead of emerging cybersecurity threats, organizations need to move toward adaptive, Tier 4 cybersecurity implementations.

Table 1.1: NIST Cybersecurity Framework Implemetation Tiers

Tier	Risk Management Process
Tier 1: Partial	Organizational cybersecurity risk management practices are not formalized, and risk is managed in an ad hoc and sometimes reactive manner.
Tier 2: Risk Informed	Risk management practices are approved by management but may not be established as organizational-wide policy.
Tier 3: Repeatable	The organization's risk management practices are formally approved and expressed as policy. Organizational cybersecurity practices are regularly updated based on the application of risk management processes to changes in business/mission requirements and a changing threat and technology landscape.
Tier 4: Adaptive	The organization adapts its cybersecurity practices based on previous and current cybersecurity activities, including lessons learned and predictive indicators. Through a process of continuous improvement incorporating advanced cybersecurity technologies and practices, the organization actively adapts to a changing threat and technology landscape and responds in a timely and effective manner to evolving, sophisticated threats.

Existing commercially available Security Information and Event Management (SIEM) systems are widely used as a first line of defense for teams like the CDC. Most often, they feature signature-based threat detection, which can identify known threats by their encoded characteristics. These methods are ineffective in protecting against unknown signatures, what are known as zero-day attacks. In some cases, SIEM systems may have anomaly detection algorithms built-in or implemented through extensions. While signature-based threat detection alerts cybersecurity analysts of the specific threat they may be facing, anomaly detection systems most often do not share context related to the type of threat or the reasoning behind the anomalous predictions. This lack of context can be particularly true when black-box machine learning models are used to create the system alerts. In anomaly detection, the lack of context can increase the time required for cybersecurity analysts to investigate and categorize the anomaly, driving down the number of threats which can be investigated by a team.

1.2 Objective

In order to improve the CDC's position in relation to zero-day attacks, the project addresses the key requirements outlined below:

1. **Resiliency:** The system must be resilient to non-stationary data and should be able to natively re-establish its own baseline so that additions, deletions, or changes to the network's structure are handled without impact to its performance and without any manual configuration.

2. **Scalability:** The system must use distributed infrastructure to be able to characterize infinitely scalable data, in order to meet the extreme throughput of over 2 billion daily records currently generated on the network.

3. **Atomicity:** The system must not rely on aggregation of data points to identify outliers, but must ensure atomicity by individually scoring each data point.

4. **Interpretability:** The system must be capable of providing interpretable anomaly scores to cybersecurity analysts, enabling visualization tools to contextualize the score in relation to other behavior on the network so that anomalies can be categorized quickly and analyst workflows can be made more efficient.

5. **Supervised Learning Annotations:** The system must have the ability to learn from cybersecurity analyst investigations, which offer valuable context towards characterizing future anomalies and should be used in closed-loop machine learning models, whose predictions must be provided on future outlier predictions to further reduce analyst workflows.

6. **Flexibility:** The system must be flexible such that it can be applied to arbitrary multidimensional data, allowing for new fields to be introduced to the model. This flexibility will also enable its use in a variety of domains outside of network traffic, including finance, retail, healthcare, government and other outlier detection use cases.

The system requirements above outline a system which can be deployed quickly and without excessive configuration, because it is resilient to non-stationary data and arbitrarily large datasets. It takes a novel approach towards anomaly detection which does not use aggregation, but benefits from both scoring and labeling atomic traffic flows. It leverages the knowledge and insight of cybersecurity analysts through their already context-rich investigations into potential threats, enhancing its own predictions by using deep human investigations to learn and improve future anomaly predictions. Finally, the requirements ensure a generalizable framework for anomaly detection in a wide variety of contexts outside of cybersecurity, including finance, retail, healthcare, government and more.

Chapter 2

Background

At their core, cybersecurity threats exploit the increased complexity and connectivity of critical infrastructure systems. As defensive tools used in industry become continuously more sophisticated, leveraging new technologies like artificial intelligence, so too do the methods used by attackers. On top of increased sophistication from attackers, cybersecurity analysts also have to defend an ever-increasing attack surface [4]. Each new device on a network creates N new connections, where N is the number of connections from the new device to adjacent attack vectors. These connections create superlinear growth in the potential vulnerabilities that an attacker can exploit. Figure 2-1 [5] shows forecasted growth of internet-connected devices through 2020.

2.1 Threat Taxonomy

Researchers have created a wide range of classifications for malicious activities on a computer network [6, 7, 8]. Malicious traffic can be categorized into three stages of an attack, which are typically sequential - Reconnaissance, Scanning, and Attacks. Reconnaissance is often the initial and most important stage for an attacker, used by malicious users to gather data on the target such as IP address range, DNS records, and mail servers. Scanning (or port scanning) is used by attackers to test the boundaries of the network and firewall, and to identify which services are running on each host. Finally, informed by reconnaissance and scanning, attacks involve an attempted

Machines Go Online
The number of everyday objects, or "things," connecting to the Internet will exceed PCs and smartphones.

Connected devices (billions)

Legend: Things | PCs & laptops | Tablets | Mobile phones
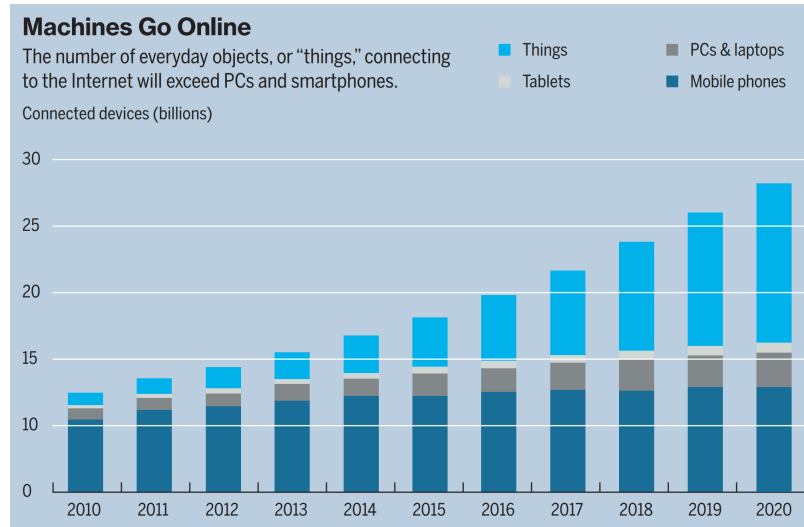
Figure 2-1: Predicted Growth of Connected Devices

exploit of the computer network.

Some attacks are used to deliberately cause damage to a system, while other attacks obtain information about a system or set of users without causing any perceptible damage. Many of the most high-profile attacks in recent years have fallen into the second category, and these data breaches are often not identified until months later. Hindy et al. classify attacks into five broad categories:

1. **Network:** Network attacks include Denial of Service (DoS) and Packet Sniffing, and are executed from outside of the network.

2. **Host:** Host threats include Malware and Viruses, which typically make their way onto a host machine through networked or physical means, and are then executed from within the host.

3. **Software:** Software threats include SQL Injection and Cross-Site Scripting (XSS), and typically relate to exposed systems (such as web applications) with vulnerable fields that are exploited by attackers.

4. **Physical:** Physical threats include physical backdoors.

5. **Human:** Human factors play a large role in vulnerabilities of computer net-

8
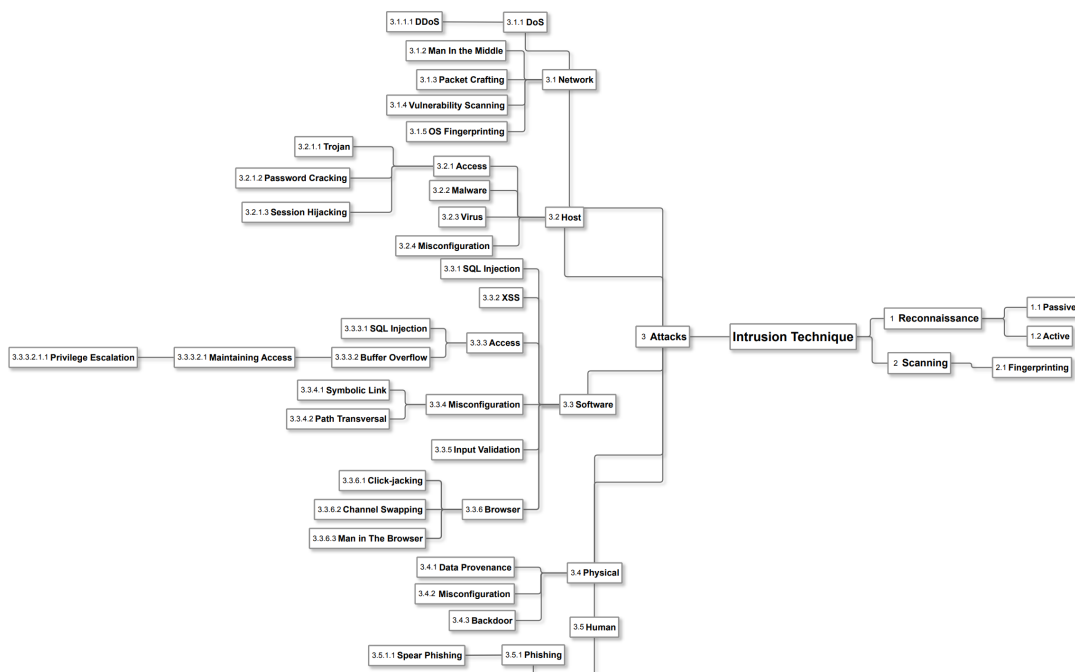
Figure 2-2: Taxonomy of Malicious Network Traffic

works. These include attacks like phishing, which exploit internal users, as well as insider threats who have access and privileges to be on the network.

In practice, the cybersecurity analyst has visibility into many of these different attacks if they manifest in changes in network behavior. For example, exfiltration of data from an insider threat would not be preempted by any perceptible reconnaissance or scanning signatures, but if the data is exfiltrated over the network, it will be captured in data passing through routers on the network.

Many organizations have their own defined categorizations of malicious network behaviors, which may be internally defined or inherited by off-the-shelf security products such as RSA Archer, LogRhythm, or other SIEM platforms. As new threats are uncovered and attackers use more sophisticated techniques, these classifications need to maintain ever-increasing categories of attacks in order to hedge off Advanced Persistent Threats (APTs). The most relevant tools in the cybersecurity landscape are those which can keep up with the bleeding edge of cybersecurity threats, and tools that cannot keep up will be relegated out of use.

9

2.2   Network Traffic Data

Traditional approaches towards detecting anomalies in network traffic data originally relied primarily on packet analysis techniques (payload-based intrusion detection). Using these techniques, cybersecurity tools such as Snort and Bro investigate the data encompassed in each individual packet sent over a network. However, as the volume of network traffic data began to grow, storage and processing requirements for payload-based intrusion detection systems became extremely large. Furthermore, over 80% of enterprise's web traffic is encrypted in 2019, according to Gartner [9], which makes packet analysis impossible in most cases.

Because of the challenges associated with packet analysis, researchers and operators began turning towards flow-based intrusion detection, which can more easily be performed at high speeds and do not require immediate investigation into the payload of network traffic. Flow-based intrusion detection, as specified in the IP Flow Information Export (IPFIX) [10] and Netflow [11] standards, aggregates a collection of packets sent between two machines over a network. Aggregated flows are represented as bidirectional flows (biflows) with a biflow source and a biflow destination, and some data associated with the transaction. Biflows include data from directional key fields, which represent data associated a single endpoint of the flow, like "Source IP", "Destination IP", "Source Bytes", or "Destination Bytes". Biflows also include nondirectional key fields, which represent data not specifically associated with either endpoint of the flow, such as "Protocol".

Biflows can also be thought of as two unidirectional flows (uniflows), representing data sent and received from each individual endpoint within the transaction. During analysis, biflows can be destructed into a forward direction (of a biflow) uniflow and a reverse direction (of a biflow) uniflow. For a given direction, directional data can be described as normal information elements if associated with the same direction (ex. Source Bytes is a normal information element of a Source), or reverse information element if associated with the reverse direction (ex. Destination Bytes is a reverse information element of a Source).
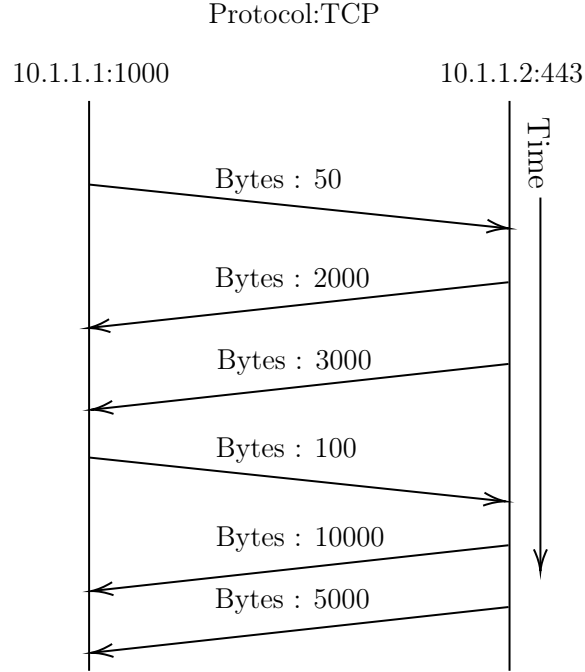
Figure 2-3: Example Traffic Pattern for Netflow Data

Figure 2-3 shows an example of data transferred between two machines. In live systems, any time a data packet (such as a Transmission Control Protocol (TCP) packet) crosses a metering flow export device, such as a router or network switch, the flow export device automatically merges the transfer with any previous data transferred between the two devices. Without Netflow aggregation, these six packets would be stored as six individual rows (two of source 10.1.1.1:1000 and four of source 10.1.1.2:443). In this case, the Netflow aggregation results in a single flow record, with source 10.1.1.1:1000, destination 10.1.1.2:443, 150 bytes transferred from the source machine, and 20,000 bytes transferred from the destination machine.

Organizations collect Netflow data from many flow export devices into a single flow collector, which can then be used to compute network statistics or for investigations by cybersecurity analysts.

In this research, data from an enterprise SIEM system was used. The SIEM exported 17 fields for analysis, shown in Table 2.1. The context provided by the SIEM was expanded to include the a single layered /16 subnet associated with each machine. By connecting additional data sources, export fields could expand to include additional directional key fields or nondirectional key fields, such as physical

11

Table 2.1: Fields available in the Enterprise SIEM Environment

Field	Description
Flow Type	Standard Flow - Bidirectional traffic
	Type A - Single-to-Many (unidirectional), for example, a single host that performs a network scan.
	Type B - Many-to-Single (unidirectional), for example, a DoS attack.
	Type C - Single-to-Single (unidirectional), for example, a host to host port scan.
First Packet Time	Specifies the date and time that flow is received.
Last Packet Time	Specifies the time that the flow is stored in the SIEM database.
Source IP	Specifies the source IP address of the flow.
Source Port	Specifies the source port of the flow.
Source Packets	Specifies the total number of packets that are sent from the source host.
Source Bytes	Specifies the number of bytes sent from the source host.
Source Flags	Specifies the TCP flags detected in the source packet, if applicable.
Destination IP	Specifies the destination IP address of the flow.
Destination Port	Specifies the destination port of the flow.
Destination Packets	Specifies the total number of packets that are sent from the destination host.
Destination Bytes	Specifies the number of bytes sent from the destination host.
Destination Flags	Specifies the TCP flags detected in the destination packet, if applicable.
Protocol	Specifies the protocol that is associated with the flow.
Category	Network traffic category
Source Subnet	A logical subdivision of the Source IP network.
Destination Subnet	A logical subdivision of the Destination IP network.

MAC addresses, users, roles, or whether connections are with an internal or external machine.
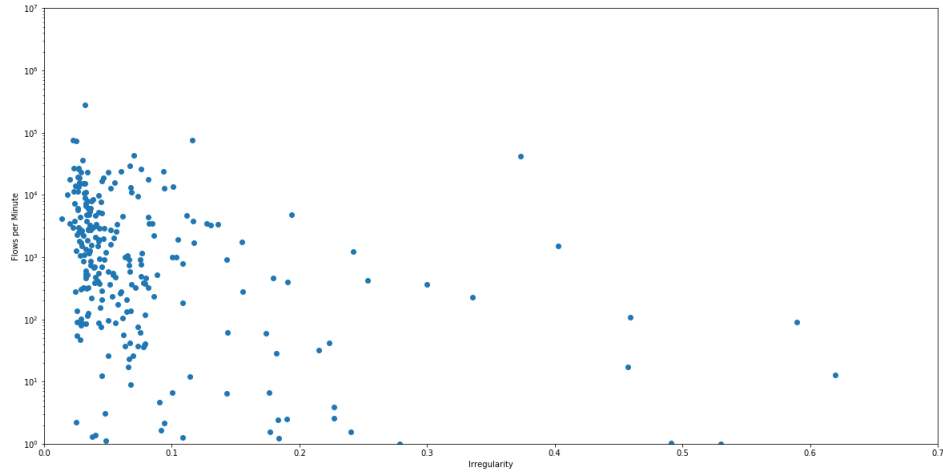
(a) All Subnets
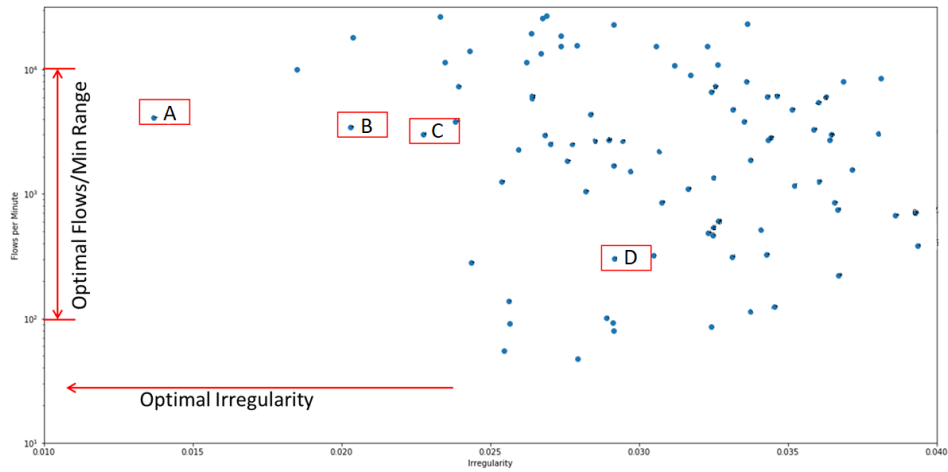
(b) Low-Irregularity Subnets

Figure 2-4: Tradespace Analysis of Subnet Size and Irregularity
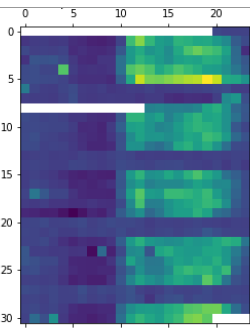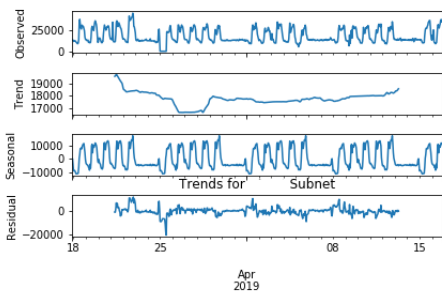
Due to the large size of the network, the 2 billion daily Netflow records resulted in approximately 250 GB of Netflow data each day. For the system design and test, it was beneficial to select a smaller subset of the Netflow data which could be used as a representative sample of the network. I used a tradespace analysis (see Figure 2-4)

to select a single /16 subnet for analysis within this project, based on optimizations of Flows per Minute and Irregularity, which is a composite of the variance of non-seasonal components after a seasonal decomposition of data transferred over each subnet. Refer to Table 2.2 for an analysis of selected subnets within the network.

Table 2.2: A Sample of Flows/Min on 4 Subnets within the CDC's Network

Subnet	Flows/Min	Heatmap	Seasonal Decomposition
A	4131		
B	3456		
C	2999		

Continued on next page

Subnet	Flows/Min	Heatmap	Seasonal Decomposition
D	303		

The final selection was made by recommendation from the CDC. The subnet features approximately 300 flows per minute, resulting in about 63 MB of data per day. The it has relatively low composite irregularity, and features highly periodic (sometimes referred to as seasonal) data. It is composed primarily of host machines and WAPs and has a clear distinction between occupied and unoccupied levels of activity, as distinguished by business hours in the time zone of the subnet.

2.3   Intrusion Detection Systems

Enterprise Intrusion Detection Systems (IDSs) (sometimes called Network Intrusion Detection Systems (NIDSs)) use Netflow data to identify intrusions on their networks. IDSs identify potential threats in one of two ways [12]:

1. **Signature-Based Intrusion Detection:** the process of comparing signatures against observed events to identify possible incidents.

2. **Anomaly-Based Intrusion Detection:** the process of comparing definitions of what activity is considered normal against observed events to identify significant deviations.

Signature-based IDSs play a significant role in keeping networks safe from APTs, because they can define patterns of known malicious behavior in order to alert cybersecurity analysts of known threats, or to directly respond and recover from an intrusion using security automation tools [13]. One simple example of a signature-based IDS

is a blacklist of discrete entities, such as hosts, TCP ports, URLs, filenames, or file extensions that have been determined to be malicious [12]. These types of threats, once identified, can be quickly encoded and distributed to security teams through their SIEM or other enterprise security tools. However, since signature-based intrusion detection is designed for identifying pre-defined intrusions, it has no capabilities to detect novel attacks.

Anomaly-based intrusion detection is a powerful tool for protecting networks from new attacks, and together with signature-based, forms a strong line of defense for security teams. It relies on a formulation of what is "typical" within the system's baseline behavior, which is usually based on prior behavior of the system. In this research, we refer to this baseline behavior as a prior. Determining what is "typical" against what is "anomalous" is the most challenging task for anomaly detection systems, and most critical to ensure high recall and low false positive rates and alarm fatigue.

This research builds on foundational concepts within anomaly detection literature, including Subject-Object Relationships, Aggregated Statistics, and Point, Contextual, and Collective Outliers. The implementation extends these concepts in order to define a generalizable system with learning capabilities which can be used with arbitrary multi-dimensional data.

2.3.1 Subject-Object Relationships

In her seminal paper on anomaly-based intrusion detection [14], Dorothy Denning used the terms *subject* and *object* to define context within an intrusion detection system. Subjects are the initiators of actions in the target system, and are sometimes divided into different classes or groups, which may overlap. Objects are the targets of actions and include programs, messages, and terminals. For example, in her meaning, a subject may be a user, or user group, and an object may be a program or file that the user is accessing. Therefore, anomalies can be detected using the context of both subjects (should **this** user be accessing that resource?) *and* objects (should this user be accessing **that** resource?)

In this research, I have extended the meaning of *subject* and *object* to fit flexibly within a generalizable framework for use in cybersecurity and other domains. In this research, a *subject* is a field used for identification of a class or group which may have representative behavior. An *object* is any other field, related to information such as resource identifiers, statistics, or other fields, which can be compared for representative behavior. For example, for the *subject* field "Source IP", the Netflow field "Destination IP" is an *object* field representing an accessed resource, and the Netflow field "Source Bytes" is an *object* field representing statistics of data being transferred. An anomaly detection system can predict anomaly scores on either accessed resources (should this Source IP be interacting with that Destination IP?) or on statistics (should this Source IP be sending that many Bytes?).

2.3.2 Aggregated Statistics

Many existing anomaly detection systems suffer from non-generalizable aggregation statistics used to identify anomalies. As an example, administrators of anomaly detection systems may choose to monitor connections between machine A and machine B, and calculate statistics such as mean and standard deviation on the amount of data transferred between the machines [15]. However, by calculating statistics between every source and destination IP address, the system will incur $\mathcal{O}(n^2)$ space complexity and the superlinear growth cannot be maintained on large computer networks. An alternative to calculating statistics across every pair of IP addresses is for administrators to manually configure which statistics are most relevant and should be captured. By introducing manual configurations, the system loses its generalizability, and also introduces potential holes where administrators have not implemented anomaly detection routines. In order to ensure scalability in growth of large networks, the system should maintain approximately $\mathcal{O}(n)$ space complexity and use little to no manually configured aggregations to identify anomalies.

Even in anomaly detection systems with linear space complexity, model designers often use aggregation methods to identify outlier data. For example, the charts from Figure 2-5 show the count of total number of network sessions for the same

17

(a) Network Sessions, Aggregated by Hour   **(b)** Network Sessions, Aggregated by Minute
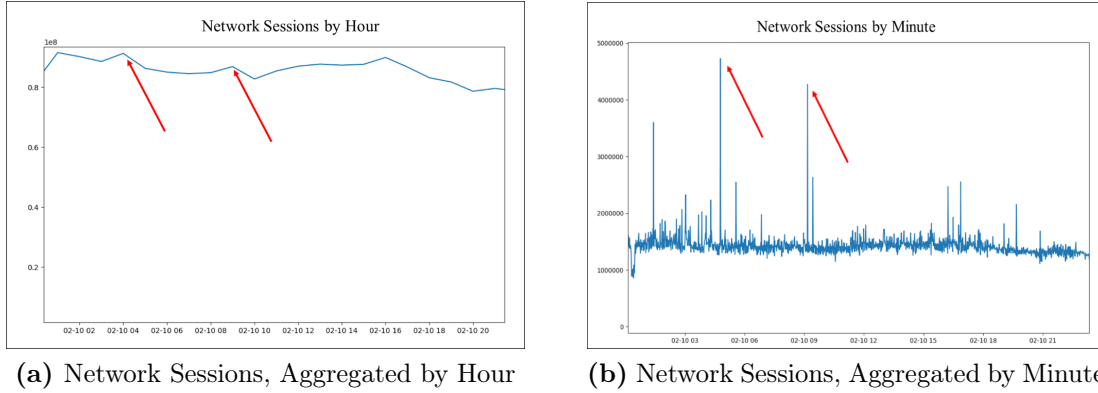
Figure 2-5: Network Sessions Aggregated by Hour, Minute

network, aggregated over both hour and minute. Some very significant spikes can be seen when aggregated by minute which are not visible at all when aggregating by hour. Moreover, of these charts had instead been aggregated by second, it would be come nearly impossible to identify outliers, because aggregated at the second level, the data is so irregular that there might appear to be outliers occurring hundreds or thousands of times per day on this particular subnet. Clearly, by using aggregation to identify outliers, anomaly detection systems create significant risk for administrators to incorrectly configure anomaly detection systems, where the aggregation timeframe is either too long and suppresses all potential outliers, or it is too short and triggers on too many occurrences.

Additionally, by introducing aggregation at any timeframe, it becomes significantly more difficult for the CDC to investigate and respond to outlier alerts. When a network includes hundreds of thousands of interactions per minute, identifying an offending interaction or set of interactions may be a non-trivial task for a cybersecurity analyst, increasing the investigation time of an outlier alert. A system which instead scores each interaction atomically removes the need for analysts to spend time identifying the offending interaction, thus increasing response efficiency.

2.3.3 Point, Contextual, and Collective Outliers

The anomalies presented in the previous section could be described as "global point anomalies", which can be distinguished as outliers against the entirety of the es-

tablished prior. This is just one of three types of outliers which anomaly detection systems need to identify [16]:

1. **Global Point Outlier:** an individual data instance that can be considered as anomalous with respect to the rest of data; the simplest type of anomaly and is the focus of majority of research on anomaly detection.

2. **Contextual Point Outlier:** an individual data instance is anomalous in a specific context (but not otherwise).

3. **Collective Outlier:** a collection of related data instances that are anomalous with respect to the entire data set. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous.

(a) Global Point Outlier   **(b)** Contextual Point Outlier   **(c)** Collective Outlier
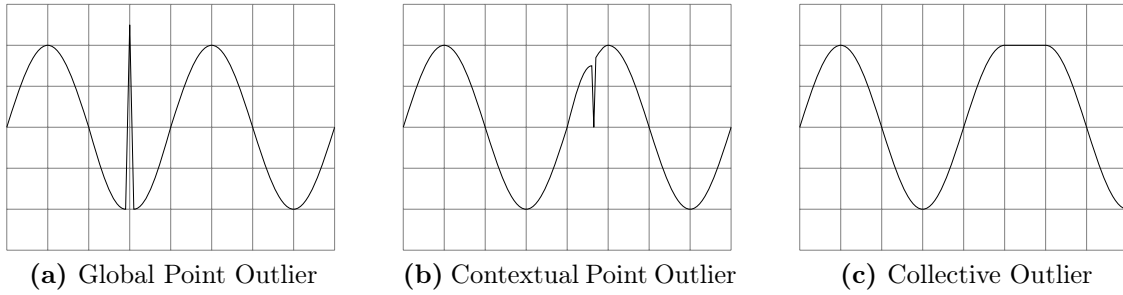
Figure 2-6: Point, Contextual, and Collective Outliers

Contextual and collective anomalies are the most difficult to identify and reason about in large interconnected networks. With this system, I focus on the first two types of anomalies; scoring individual data points is conducive to atomic annotations and supervised learning algorithms; an effective anomaly detection system should be able to identify collective outliers as individual outliers with a high recall, and score each individually.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Methodology

Towards the goal of an interpretable and scalable system for detecting real-time anomalies in multidimensional streaming data, I present in this section the Ensemble Outlier Detection System, a two-step methodology for baselining network behavior, calculating anomaly scores for Netflow data points, and interpreting their results. I then present a processing architecture to ensure continuous throughput of data, so that baseline behavior is continuously updated for accurate up-to-date anomaly predictions. The processing architecture uses the concepts *window length* and *sliding interval* found in other streaming frameworks to periodically remove outdated data from the baseline model.

3.1 The Ensemble Outlier Detection System

The strategy used in this thesis for detecting traffic anomalies uses the concept of ensemble methods for outlier detection. The ensemble method uses a collection of individual predictions to produce a consensus prediction indicating anomalousness of each individual network interaction [17]. In this case, predictions are calculated for each object field, which is based on the context provided by subject fields. The ensemble process is shown in Figure 3-1.

This process maximizes the contextual information available for each object field while assuming covariance between objects to be zero. This assumption allows object
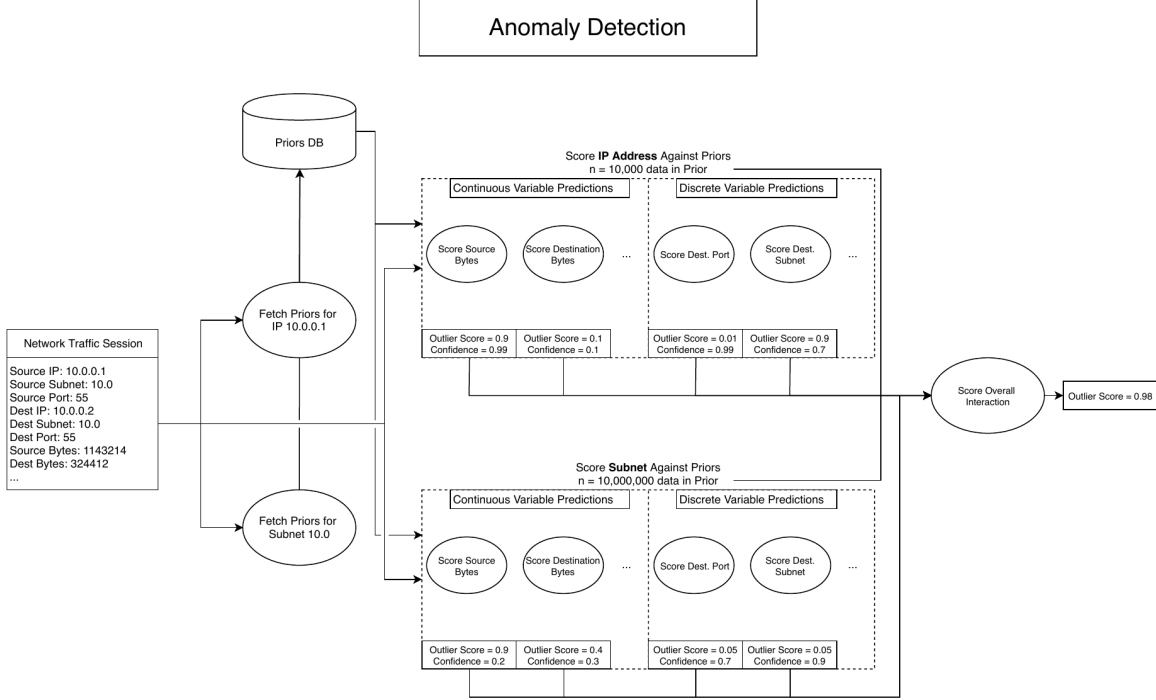
Figure 3-1: The Ensemble Outlier Detection System

variables to be treated as independent, thereby enabling a scalable design with a space complexity of approximately $\mathcal{O}(n)$.

3.1.1 Object Predictions

The ensemble-based anomaly detection methods used in this research rely on combining predictions from an arbitrary number of independent data fields. In order to combine data fields, this research generates two values, **score** and **confidence**, for each available data field. Together, these values are combined into the object's anomaly prediction.

Data Types

In network traffic, financial, and other domains, the multivariate data points representing a single interaction often take the form of a variety of primitive data types, including integer, floating-point, string, boolean, and timestamp. The predicted anomaly score of numeric data types typically uses intermediate state such as mean

or standard deviation from the mean, but these statistics do not hold for other data types. Therefore, it is necessary to generate unique methods for calculating score and confidence of each unique data type.

In this research, I have defined four priors data types and requisite score and confidence methods for each. The four priors data types are:

1. **Numeric:** integer, floating-point

2. **Categorical:** string, certain numerics (such as Network Port)

3. **Binary:** boolean

4. **Timestamp:** timestamp

Some data types, such as categorical and boolean, share certain characteristics, and in those cases functionality is shared between both data types. However, because the flexibility of scoring non-numeric data types introduces some new, differing calculations between different data types, the potential of unequal weighting of data types is also introduced. In order to mitigate the effects of unevenly weighted data types, I have introduced a tuning parameter called *scaler* into each individual score and confidence function to allow for re-scaling of any parameter.

In designing priors data structures, each has been designed for low storage requirements, low computation requirements, and interpretability in comparison to new observations. Ideal outlier predictions exhibit the following characteristics:

- Higher outlier prediction scores when scored against tight prior distributions

- Higher outlier prediction scores when scored against distributions with more prior observations

- Lower outlier prediction scores when the observation has been seen before in the subject's context

- Lower outlier prediction scores for values which are commonly observed

Timestamp data types in particular exhibit interesting characteristics in terms of seasonality. In this research, I have chosen not to fully implement priors on timestamp fields at this time; the future work will be discussed in Section 5.2.

Prior Data Structure

Each individual object can be assessed against priors describing how that particular subject has behaved before. The priors structure is designed to represent the normal boundaries of behavior for the given data type, with particular attention paid to the outer limits of those boundaries.

Priors for numeric variables, shown in JSON Snippet 3.1, store standard numeric aggregations, including count, minimum, maximum, mean, and standard deviation. They also include a Cumulative Distribution Function (CDF), and persists actual CDF values, with a denser cluster of values close to the lower and upper boundaries. The standard aggregations and CDF allow for inference around how tightly distributed the historical values have been, as well as how much data exists for the object, which inform the object's confidence. The CDF, along with aggregations, informs the score for new data points which lie near or beyond the boundary of previous behavior, and how unexpected the new data point is with respect to historical data.

Priors for categorical variables, shown in JSON Snippet 3.2, also store count of fields, but require additional computation to identify analogous statistics to those in the numeric prior. In categorical data, the CDF is nothing but a count of unique values which the subject has encountered previously. This will be shown in section 3.1.1 to be analogous to the numeric CDF; both can be inverted into a Percent Point Function (PPF) in order to score how a new data field compares to previously encountered data points.

In order to create a reference point for the inherent variability within a categorical prior, I generated a new metric called `category_equivalent_stdev`, which is analogous to standard deviation and measures how tightly packed a given categorical distribution is. Category equivalent standard deviation takes the form of a standard deviation of a series of n points, where n is the length of the prior, whose value cor-

```
1   {
2      "mypackets":{ #numeric_field
3         "prior_length":737,
4         "nan_length":0,
5         "mean":25.0,
6         "stdev":33.0,
7         "cdf":{
8            "0.0":0.0,
9            "1.52587890625e-05":0.0,
10           "0.00390625":0.0,
11           "0.015625":0.0,
12           "0.0625":5.0,
13           "0.25":12.0,
14           "0.5":15.0,
15           "0.75":25.0,
16           "0.9375":35.0,
17           "0.984375":55.0,
18           "0.99609375":150.0,
19           "0.9999847412109375":175.0,
20           "1.0":600.0
21        }
22     },
23     "otherpackets": {...}
24  }
```

JSON Snippet 3.1: Priors Structure for Numeric Variables

responds to some distance Loc_n above zero. All points of the same cdf item have the same Loc_n when computing the category equivalent standard deviation. Beginning with the item with the largest count in the cdf, each point has a Loc_n of zero. Continuing in descending order, each new item receives a Loc_n which increments by some static number `category_equivalent_distance` between 0 and 1, which is shown in Equation 3.1.

$$Cat.Eq.Dist = 1 - \frac{\max(cdf)}{prior\_length} \qquad (3.1)$$

```
1   {
2       "otherip": {  #categorical_field
3           "prior_length": 737,
4           "nan_length": 0,
5           "category_equivalent_stdev": 1.06105,
6           "category_equivalent_distance": 0.264,
7           "cdf": {
8               "208.80.152.201": 557,
9               "74.125.65.91": 101,
10              "74.125.157.99": 31,
11              "69.171.224.11": 20,
12              "72.21.211.176": 20,
13              "207.97.227.239": 4,
14              "199.59.149.230": 2,
15              "151.101.65.69": 2
16          }
17      },
18      "othersubnet": {...}
19  }
```

JSON Snippet 3.2: Priors Structure for Categorical Variables

$$\forall n \leq len(cdf): \ cdf_n \leq cdf_{n-1}, \ Loc_{cdf_n} = \begin{cases} 0 & n = 1 \\ Loc_{cdf_{n-1}} + Cat.Eq.Dist & n > 1 \end{cases} \quad (3.2)$$

In JSON Snippet 3.2, the category equivalent standard deviation $Cat.Eq.StDev$ will be composed of 557 points with a value of 0, 101 points with a value of $Cat.Eq.Dist$, 31 points with a value of $2 * Cat.Eq.Dist$, etc. The metric was designed so that CDFs with only 2 unique values have a lower category equivalent standard deviation than CDFs with 3 unique values, and 2-value CDFs with a 50-50 split will have a higher category equivalent standard deviation than CDFs with a 95-5 split.

Finally, priors for binary variables, shown in JSON Snippet 3.3, are similar to those for categorical variables, but use the concept of *binary mean* as opposed to *categorical standard deviation.*

```
1   {
2       "is_source": {   #binary_field
3           "prior_length": 737,
4           "nan_length": 0,
5           "binary_mean": 0,
6           "cdf": {
7               "false": 737
8           }
9       },
10      "protocol": {...}
11  }
```

JSON Snippet 3.3: Priors Structure for Categorical Variables

Binary distributions are special cases of categorical distributions, whose mean will always be between 0 and 1, and whose relative distribution can be calculated as the mean's distance away from 0.5. If a binary mean is 0.5, (distance from 0.5 = 0), the underlying distribution can be said to be perfectly symmetric. If a binary mean is either 0 or 1 (distance from 0.5 = 0.5), the underlying distribution has no variation. I exploit this property when calculating anomaly scores, and thus add the field `binary_mean` for calculating anomaly scores in the outlier prediction portion of this research.

Score Components

In this research, score is used to compare the value of a given field to its expected behavior based on the subject's prior. The calculation for score differs for each of the four priors data types, but in every case the score is composed of a collection of score components. Each score component represents an additive dimension which distinguishes anomalies from expected behavior, and takes some value between 0 and 1. Score is calculated as the product of all score components for a given priors data type, as seen in Equation 3.3.

$$Score = \prod_{n=1}^{len(SC)} ScoreComponent \qquad (3.3)$$

$$Score_{numeric\_ppf} = [2|(x - 0.5)|]^n \qquad (3.4)$$

$$Score_{dist\_from\_mean} = \frac{\sqrt[n]{x+1} - 1}{\sqrt[n]{x+1}} \qquad (3.5)$$

$$Score_{categorical\_proportion} y = (1-x)^n \qquad (3.6)$$

$$Score_{novelty} = \begin{cases} 1 - \frac{0.5x}{x+\left(\frac{1}{n}\right)} & category \notin prior \\\\ \frac{0.5x}{x+\left(\frac{1}{n}\right)} & category \in prior \end{cases} \qquad (3.7)$$

Representations of the score components in this research are shown in Figure 3-2. Each score component contains a tunable parameter `scaler` (represented as n) which defines a concrete, continuous function from a class of functions described by each score component. The `scaler` is initialized to a static value, but is intended to allow for re-optimization of its data over time. Depending on the context of the score component, concrete functions operate in the range of either $\{0 \to 1\}$ or $\{0 \to \infty\}$.

The score of numeric data types is computed using the Numeric PPF component (Figure 3-2a) and the Distance from Mean component (Figure 3-2b). The Numeric PPF component inverts the prior's CDF to compute the percent point function of a value being scored. It ensures that only numeric values which fall outside (roughly) the 5th and 95th percentiles are recognized as potentially anomalous, with exponential growth as the value approaches either 0 or 1. This is combined with the Distance from Mean component, which ensures that values farther from the mean will have a larger outlier score than those near the mean. Take for instance a prior whose maximum value was 1,000. Both of the values 1,001 and 1,000,000 will lie outside of the boundaries of the Numeric PPF (and thus receive a score of 1 for the Numeric PPF component), but intuitively the data point with a value of 1,000,000 is more likely to be a true anomaly than the outlier with a value of 1,001. By combining both the Numeric PPF component and the Distance From Mean component, the overall

28
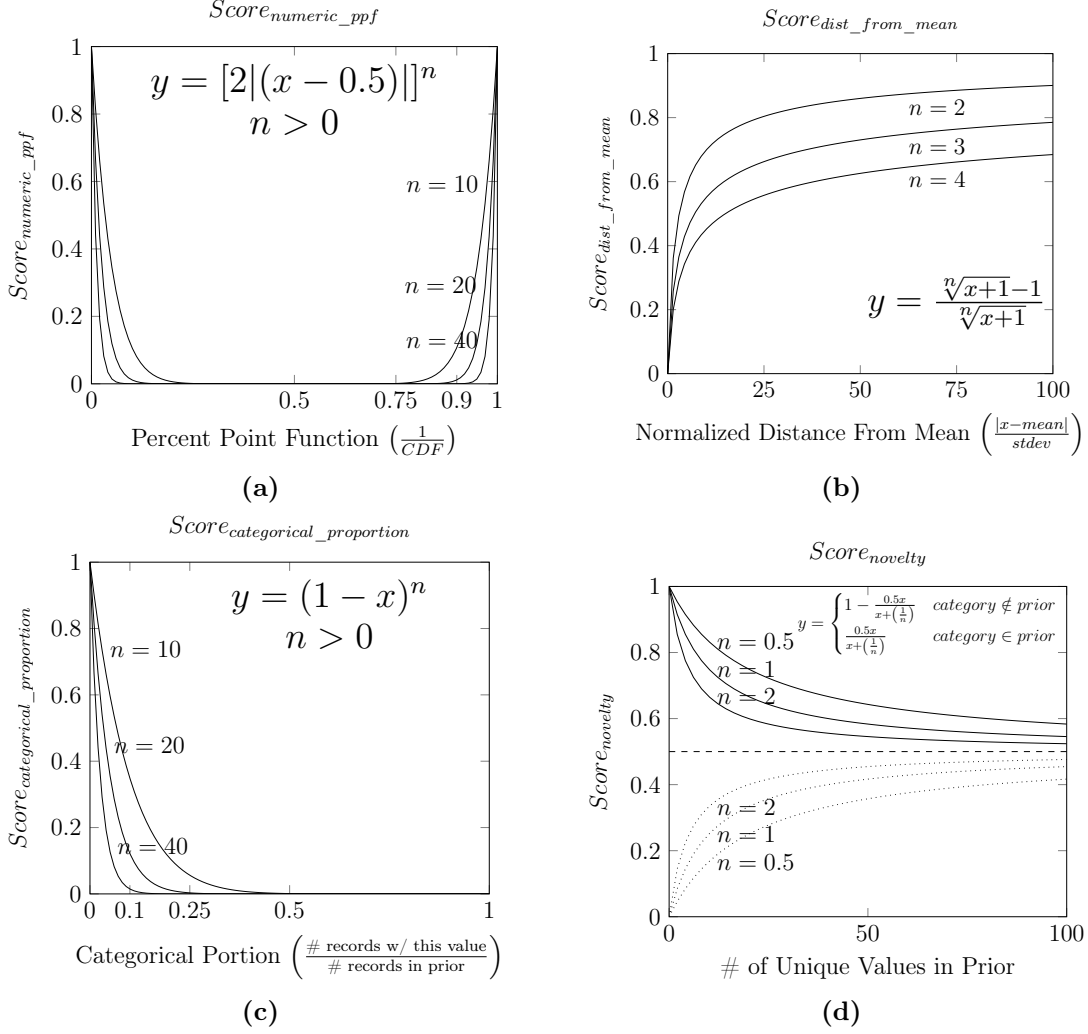
Figure 3-2: Score Components

outlier prediction of a data point scores all potential points continuously.

The score of categorical data types is computed using the Categorical Proportion component (Figure 3-2c) and the Novelty component (Figure 3-2d). The Categorical Proportion component ensures that values which have previously been encountered frequently in the prior are scored near zero, while values which have occurred very infrequently or never are scored near one. However, some networks regularly encounter traffic from new IP addresses (certain web servers, for example), while other machines regularly interact with only a few other machines. The Novelty component handles these cases by scoring the novelty high if a value is new and there are very few unique values in the prior, and lower if there are many unique values in the prior. Moreover,

if the value is not new, and there are very few unique values in the prior, it scores the value low, because priors with low unique leave little uncertainty in interpretation. However, if there are many unique values in the prior already, there is an uncertain baseline as to what is normal and what is anomalous. If the value has been seen before, the score cannot be extremely high, but it could contribute a small score in the later combination phase because of its uncertainty.

Finally, the score of binary data types is computed with both the Categorical Proportion component (Figure 3-2c), as well as the Novelty component (Figure 3-2d), since it exhibits the same categorically referenced data as categorical data types. Because binary variables have at most two categories, they do not encounter the same variability in encountering new values as categorical data does. Therefore, the scaling factor for binary data types can be increased in order to trigger binary-type anomalies at similar rates as numeric-type and categorical-type anomalies.

Confidence Components

Unlike the score parameter, confidence is used solely to characterize the predictability of the prior itself. In other words, when generating an outlier prediction for a new data point, the confidence components of the outlier prediction are not influenced by the new data point; they are instead influenced only by the associated priors. In general, priors that have little variation in previously encountered values have high confidence, while priors with a high degree of variation have low confidence. This translates to low standard deviation in numeric data fields, or one/few unique value(s) in categorical data fields. The concrete functions of confidence components operate in the range of either $\{0 \rightarrow 0.5\}$, $\{0 \rightarrow \infty\}$ or $\{1 \rightarrow \infty\}$.

$$Conf = \prod_{n=1}^{len(CC)} ConfComponent \tag{3.8}$$

$$Conf_{num\_records} = \frac{\sqrt[n]{x} - 1}{\sqrt[n]{x}} \tag{3.9}$$

$$Conf_{coef\_of\_var} = 1 - \frac{x}{x + \left(\frac{1}{n}\right)} \tag{3.10}$$

$$Conf_{category\_equivalent\_stdev} = 1 - \frac{x}{x + \left(\frac{1}{n}\right)} \tag{3.11}$$

$$Conf_{binary\_mean} = (2x)^n \tag{3.12}$$

Like score, confidence is calculated as the product of all confidence components for a given priors data type, as seen in Equation 3.8. Representations of the confidence components in this research are shown in Figure 3-3.

The confidence of every data type invariably relies on the Number of Records component (Figure 3-3a). Take for example two datasets with numeric values, with mean of 100 and standard deviation of 0 (i.e. each value == 100). Dataset 1 holds 5 values, all 100, while dataset 2 holds 1,000,000 values, all 100. If the next observed value is 105, while we may or may not raise concerns for dataset 1, the administrators of dataset 2 almost certainly have reason for severe concern. This property holds generally in the case of any data type.

Numeric data types have an additional confidence component, Coefficient of Variation (Figure 3-3b), which uses the coefficient of variation statistic CV ($CV = \sigma/\mu$) to infer the relative variability of a dataset, normalized by magnitude. For example, a standard deviation of 1,000 on a dataset whose mean is 100 may represent a lot of variability, while the same standard deviation on a dataset whose mean in 1,000,000 actually represents quite a close cluster of points.

Categorical data types use the `category_equivalent_stdev` value described earlier in this section. Similar to the Coefficient of Variation component for numeric data types, this represents a normalized variation of a given distribution, and is used in combination with the Number of Records component to describe confidence of scoring
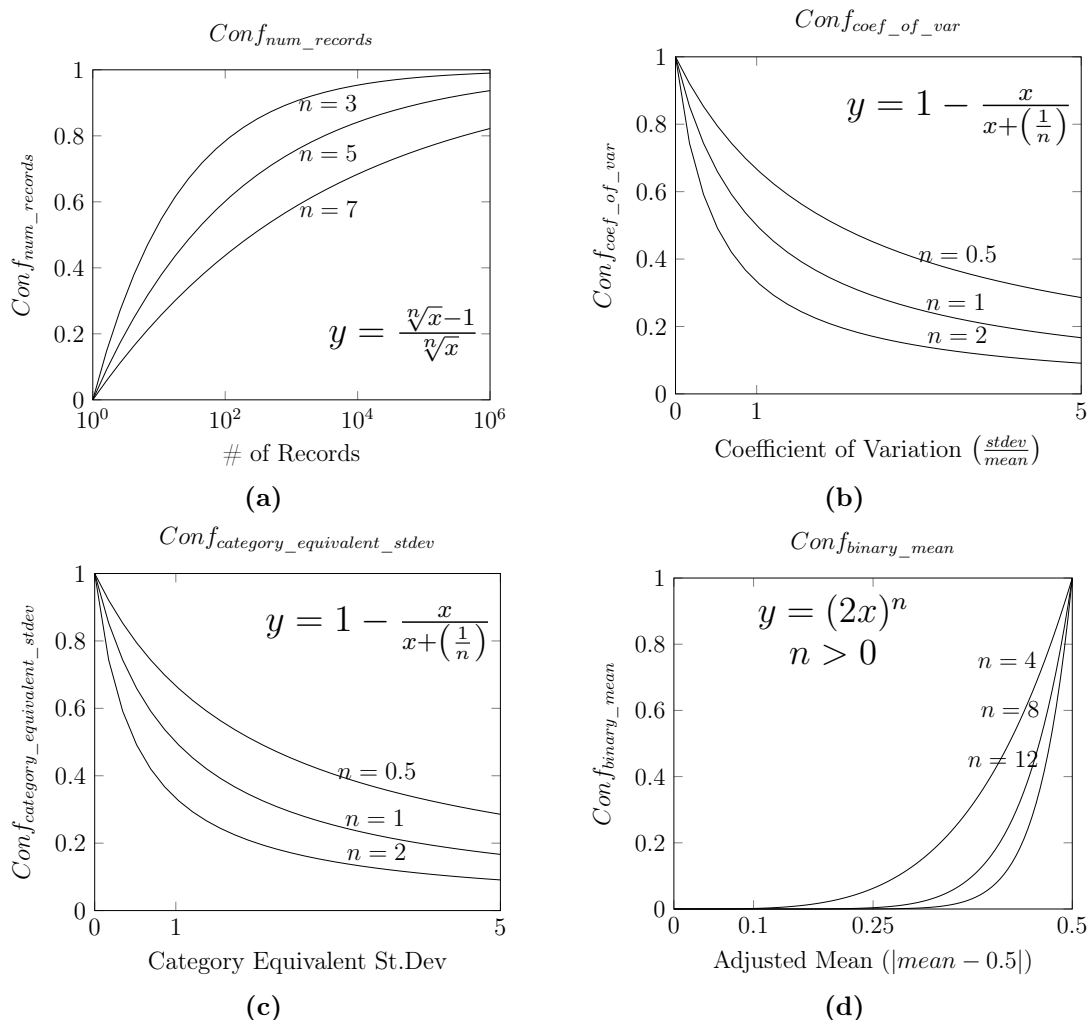
Figure 3-3: Confidence Components

a new value against the given prior structure.

Finally, binary data types use the `binary_mean` field as described earlier in this section. An intermediate variable $AdjustedMean$ is calculated as the binary mean's distance from 0.5 ($AdjustedMean = |0.5 - BinaryMean|$). For distributions whose a binary mean is near 0.5, the confidence is low, and incoming values cannot be confidently scored as anomalous on this information alone. However, as the $AdjustedMean$ approaches 0.5, the confidence of the expected value increases with a polynomial growth rate towards 1. By combining the high confidence with what is known about the value itself, we reach an appropriately low prediction if confidence is high, and the observed value matches a frequently encountered value, and an appropriately high

prediction when the observed value matches a not-frequently-encountered value.

Combining Score and Confidence

Once a given object's score components and confidence components have been combined into requisite score and confidence values, a prediction value is generated by the formula given in Equation 3.13.

$$Prediction = Score^{ScoreWt} * Conf^{ConfWt} \tag{3.13}$$

Here, again, the structure of the equation was chosen in order to reasonably suppress alerts, as power functions taken on values less than one will drive the result towards zero. The choice to use weight values $ScoreWt$ and $ConfWt$ uses the same logic for flexibility as the implementation of the `scaler` factors, as they can be used for tuning of any individual dataset as time progresses. For the sake of simplicity, in the implementation of this project I used a static value for $ConfWt$ and optimized the value of $ScoreWt$. The optimization method is further described in Section 4.

3.1.2 Combining Object Predictions

Because of the requirement to maintain atomicity of each data point, the ensemble method increases the robustness of prediction scores by individually scoring each data field based on available context. In this way, each biflow datapoint scores each of the data fields from Table 2.1 as an individual object, and uses an aggregation function of the form $Pred_{biflow} = f(Pred_{obj\ 1},\ Pred_{obj\ 2},\ ...,\ Pred_{obj\ n})$ to score the flow from the collection of object fields.

In Section 3.1.1, I demonstrated the methods used to create individual predictions for each object field within the biflow data point. The methods ensure that fields whose values are within the expected boundaries of behavior have $Pred_{obj}$ values suppressed towards zero. Suppressing expected values towards zero enables a generalizable method for scoring data points which uses recursive aggregation and

behaves identically irrespective of the number of data fields within each data point. Equation 3.14 shows the recursive aggregation function.

$$
Pred_{obj_n} = \begin{cases} Pred_{obj_n} & n = 1 \\ Pred_{obj_{n-1}} + (1 - Pred_{obj_{n-1}}) * Pred_{obj_n} & n > 1 \end{cases} \tag{3.14}
$$

Equation 3.14 is an unordered (commutative) recursive aggregation function, which allows for consistent behavior across any collection of prediction values. This property will be useful in later parts of this section.

For example, consider a simple system which scores two fields, `destinationip` and `destinationbytes`, and two potentially anomalous data points. In data point 1, the value of `destinationip` is considered normal and its prediction is 0, but the value of `destinationbytes` is abnormal and its prediction is 0.75. In data point 2, both `destinationip` and `destinationbytes` are somewhat abnormal and are scored as 0.5 each. The recursive aggregation will score each of the two data points as 0.75. The intuition is that a data point with possible outlier behavior in many of its data fields should be investigated just as data points with near-certain outlier behavior in just one data field.

Subject Hierarchies

As discussed earlier, these methods rely on a defined subject-object relationship to calculate and store priors for future predictions. In the case of the data fields described in Table 2.1, using IP Address as the subject of anomaly predictions allows for the behavior of each object field to be stored as a prior for each unique IP address.

However, under this situation, a problem arises when the network being monitored exhibits non-stationary behavior. If the network grows and a new IP address is added, the anomaly detection system will have no context which it can use to reason about the behavior exhibited on the new IP address. In order to make the anomaly detection system resilient to non-stationary data, we can instead use an inherent hierarchical

subdivision of fields in the data point, and store priors at each hierarchy. Figure 3-4 shows a potential hierarchical breakdown which could define the layers of subjects used to calculate prediction scores.
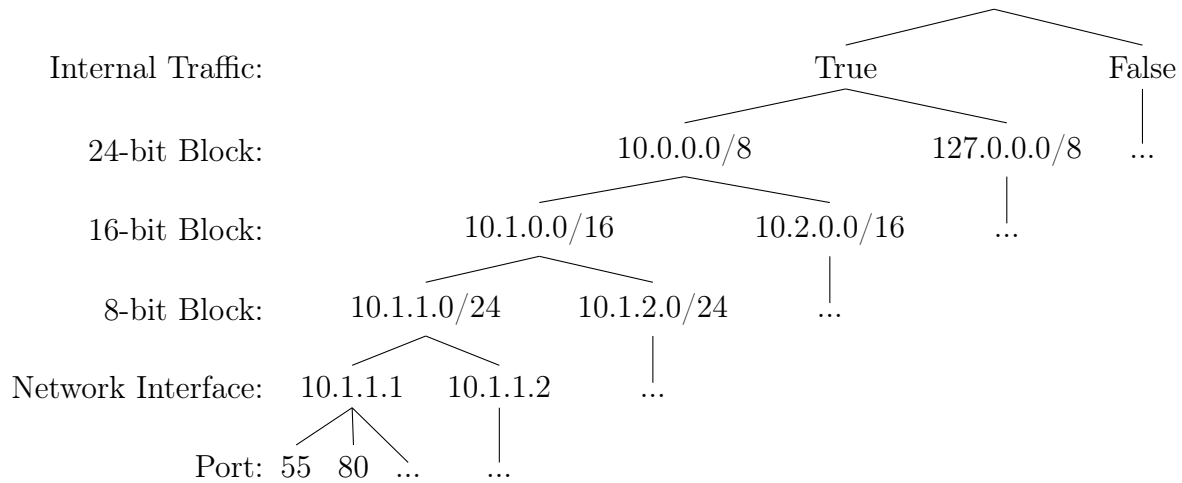
Figure 3-4: Generic Network Hierarchy for Prior Probability Segmentation

Creating a hierarchy of subjects enables less system configuration, and more generalizability, by removing the need for calibration or intuition of subject matter experts to determine which field is the most appropriate layer to choose as the subject. At higher layers of subject aggregation, there may be far too much variation to assume confidence in expected values. At lower layers of subject aggregation, data may be too sparse to assume confidence. Different domains, different networks, or even different sections of the same network may have inherently more or less confidence in each layer of the subject hierarchy, and by creating a hierarchy and predicting at each layer, the anomaly detection system can inherently generalize to capture anomalies on any network.

Additionally, higher layered subjects can treat lower layers as objects, thus expanding the potential information used to characterize the data point. Using the example of the network's new IP address, this would probably be characterized as an anomaly if the subnet's prior shows that only two distinct IPs have ever operated on the subnet. However, if the subnet's prior shows that there are hundreds of distinct

IP addresses on the subnet, then a new IP may not be a significant deviation from the expected behavior.

In this research, I implemented a two-layered hierarchy including the 16-bit subnet block and the IP address to prove the multi-layered concept. The system's space requirements increase linearly with the number of unique subjects whose priors require storage, and thus, systems with many levels of hierarchies (down to the port level or equivalent) may be required to maintain magnitudes more priors than systems with granularity at the IP level.

Like object predictions, subject hierarchies rely on the recursive aggregation function defined in 3.14 to combine subject outlier predictions into the ensemble prediction.

Biflow Directions

In the case of biflow datasets, if the subject is a directional key field, then it can exist as either the biflow source or biflow destination subject. In this case, the context can be increased by duplicating the biflow data point into two uniflow data points, one in the forward direction and one in the reverse direction. The directional key fields of the biflow destination will become reverse information elements when converted to uniflow, while the keys and values of nondirectional key fields will be the same for both new uniflow data points. In anomaly detection domains without biflow data, source data is already in a uniflow format, and thus, the conversion from biflow to uniflow is void.

In addition to existing data fields, when converting from one biflow data point to two uniflow data points, a boolean data field is added to each resultant data point called `is_source`. This flag stores the additional context available in biflow data, which can be calculated as other fields to determine a prediction score, based on whether the subject's prior is typically the source or destination of a biflow.

Similar to combinations of subject hierarchies, the prediction scores of either direction of a biflow data point are combined recursively using Equation 3.14. By using subject-object hierarchies, and by splitting biflows data points into two uniflow

data points, each data field can generate up to $2n$ unique contexts to for anomaly prediction, where n is the number of layers in the defined subject hierarchy.

3.1.3   Outlier Data Structures

By structuring the prediction scoring using subject-object relationships, treating objects as independent, and enumerating subjects by subject hierarchy and source/destination directions, a contextual tree of prediction scores emerges, which by using Equation 3.14 can be combined into a single prediction score for the entire flow. This method is resilient to missing data - for example, when only the biflow source has priors, and thus no prior is generated for the biflow destination. An example contextual tree is shown in Figure 3-5. Partial anomaly scores are available at every level from Subject Hierarchy upwards.
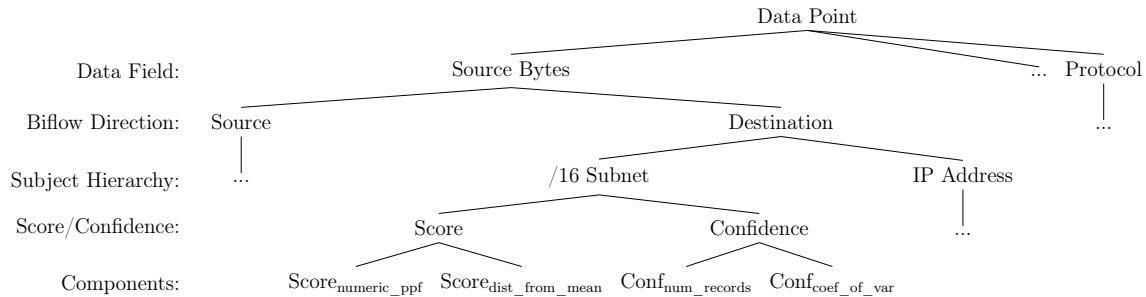
Figure 3-5: Contextual Tree for Anomaly Detection

Two primary benefits emerge from using a collapsible tree structure for overall prediction scores. First, the structure allows a cybersecurity analyst to quickly interpret high outlier scores in a given data point. With prediction scores at every level, the analyst can quickly determine if there is one or many anomalous fields, and drill down to the individual score/confidence components. The entire disaggregated outlier score of a single data point, including each of its numeric, categorical, and binary object scores at every hierarchy level and direction, is stored as a single nested data structure as shown in JSON Snippet 3.4. This enables the creation of interfaces to explore outlier predictions and compare observed values to subject priors for context

```
1    {   "prediction":1.0951643748665777e-05,
2        "objects":[{
3            "prediction":0.0,
4            "id":"sourceport",
5            "value":47806,
6            "subjects":[{
7                "id":"destinationsubnet",    #subject_hierarchy==subnet
8                "prediction":0.0,            #subject_direction==dest
9                "score":0.0,
10               "score_components":{
11                   "categorical_proportion":{
12                       "value":0.0,
13                       "scaler":40},
14                   "novelty":{
15                       "value":0.6666666666666667,
16                       "scaler":2}},
17               "confidence":0.10314973700795015,
18               "confidence_components":{
19                   "category_equivalent_stdev":{
20                       "value":0.5,
21                       "scaler":1},
22                   "num_records":{
23                       "value":0.2062994740159003,
24                       "scaler":3}}},
25               {  ...  }]}
26           {  ...  }]}
```

JSON Snippet 3.4: Data Field-level Prediction Map

of expected behaviors and deviations from those behaviors. These interfaces will be further discussed in Section 4.

By using a collapsible tree structure, an additional benefit emerges. By disaggregating the data point scores at the data field level, one can export a flat 2-dimensional data structure conducive to machine learning applications. A simple version of this structure would include the observed value of each data field, plus the prediction score of each data field, as shown in JSON Snippet 3.5. This map may be used to train future classification models. Take for example two anomalies predicted from different regions of the network. Similarities may emerge from anomalies detected on different

```json
1   {   "prediction": 0.765,
2        "sourceip": { "value": "10.0.0.1",
3                      "pred": 0.03 },
4        "destinationip": { "value": "127.0.0.1",
5                           "pred": 0.05 },
6        "sourceport": { "value": 20,
7                        "pred": 0.9 },
8        "destinationport": { "value": 21,
9                             "pred": 0.02 },
10       "<remaining fields>": {...}}
```

JSON Snippet 3.5: Data Field-level Prediction Map

regions of the network, but who saw high prediction scores in the field `sourceport`, where both values were 20 (the default FTP port). This will be further discussed in 5.2 as part of a future body of work.

3.2   Processing Infrastructure

Maintaining real-time awareness of anomalies on a large network requires two processing tasks to operate. First, the baseline must be established in the form of priors for all potential subjects on the network. In order to update priors to take new data into account, the processing infrastructure needs to be able to incrementally update the prior as new data arrives. Additionally, in order to re-establish baselines when the network's underlying structure changes, the priors processing requires some mechanism to remove outdated data after some period of time. Based on the generated priors, the system then needs to process a stream of individual data points to generate outlier predictions for each value.

3.2.1   Prior Processing Infrastructure

In order to maintain relevant baselines for outlier predictions, the generated priors should incorporate data from a recent window of time. A naive approach to generating priors for new time windows would be to retrieve all of the data for a given time

interval, calculate relevant statistics such as mean, standard deviation, and CDF, and save these these priors in persistent storage. There are two major inefficiencies with this process. First, by calculating these aggregation statistics for an entire set of data at once, as the window length begins to increase, the system's physical memory becomes a constraint. Priors are optimized for space efficiency, but the space requirements of the raw data can quickly exceed the physical system's memory, effectively limiting the window length that a system may choose. A second inefficiency also arises; in order to have the most accurate priors available, priors should be re-computed frequently. When priors are re-computed frequently, and the window used to calculate the prior is long in proportion, the system is effectively duplicating a significant portion of its prior computation each time a new prior is generated.

Windowed Operations

In order to reduce these inefficiencies, I use the concept of windowed operations from Apache Spark Streaming [18], which defines *window length* as the length of time by which an aggregation is computed, and *sliding interval* as the length of time between newly generated aggregations. The windowed operation concept is displayed in Figure 3-6. I consciously chose not to implement Spark itself due primarily to limitations in its available aggregation functions for CDF.
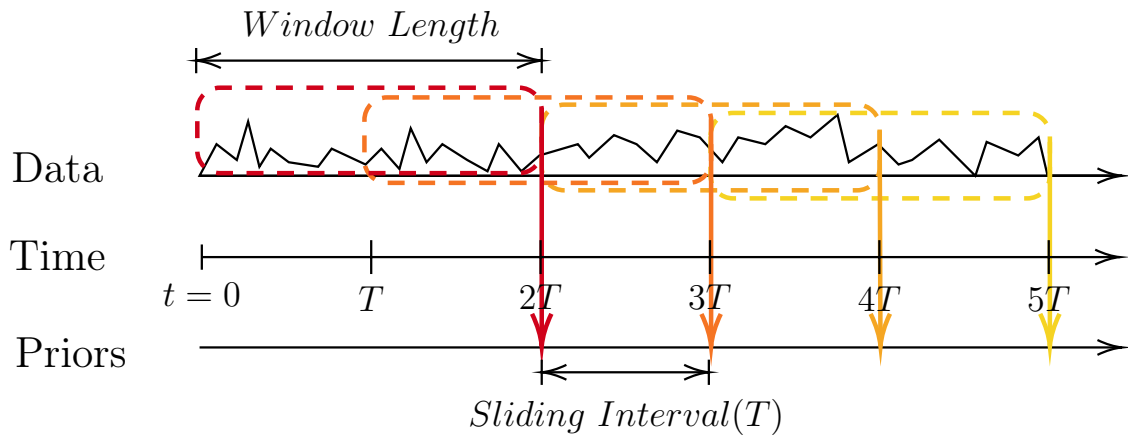
Figure 3-6: Windowed Operations

40

Using windowed operations, a new prior is generated after each sliding interval T. Therefore, the space requirements for storing historical priors is inversely proportional to the length of the *sliding interval*, and is directly proportional to the *window length*. The outlier prediction process in Section 3.2.2 references the most recent prior available when generating a prediction score for a given data point.

In order to enable windowed operations, the system must maintain some isolated *internal state* about each sliding interval for each prior. The internal state is unique to the statistic which is being calculated, such as mean, numeric CDF, and categorical CDF. Each of these internal states must implement three operations as described in the Apache Spark UserDefinedAggregationFunction documentation [18]. The internal state must implement *update*, which updates the current internal state with a single additional data point. It must also implement *merge*, which merges a pair of internal states into a single internal state. Finally, it implements *evaluate*, which generates the desired output of an aggregation function, given its internal state. By using windowed operations to calculate priors, the following process emerges:

1. **Update Recent Sliding Interval:** All new data points operate only on the internal state of the most recent sliding interval, leaving previous sliding intervals static

2. **Merge and Evaluate All Sliding Intervals:** At the end of a given sliding interval, the system calls *merge* to join together past intervals into the length equal to the window length, then calls *evaluate* to generate and store a prior for the given window length

3. **Trim Unused Sliding Intervals:** As historic sliding intervals fall out of the reach of the window length, they can be removed, freeing memory for future data

MapReduce Architecture

Through this implementation, each statistic is developed as an online algorithm, which is processed as new data arrives into the ecosystem. The project uses the

MapReduce programming model, build on top of commodity distributed servers, to design and implement processing infrastructure for generating priors. By using a MapReduce model, we implement a framework which can scale to meet the processing requirements of large anomaly detection systems. In order to ease deployment and development, in this implementation, the master node holds all configurations and routing decisions. Other expanded architectures are discussed in Section 5.2. A diagram of the system architecture as implemented is shown in Figure 3-7.
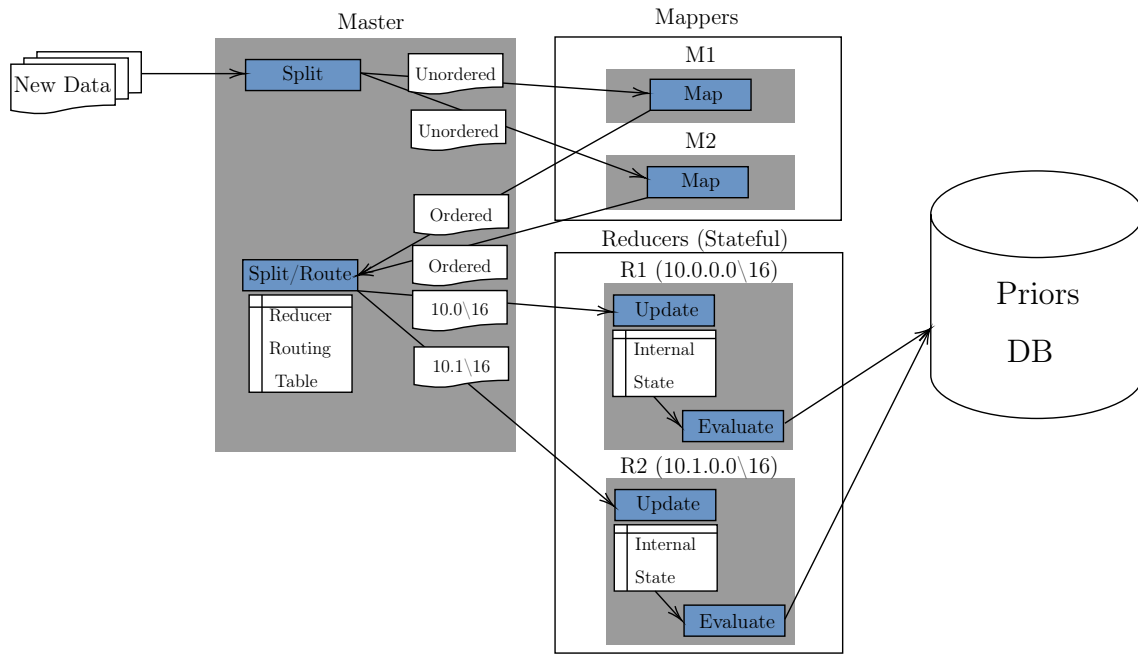
Figure 3-7: Priors Processing Architecture

Generally, new Netflow data is sent to the master and naively distributed to the stateless mappers in the system. The mapper's job is to convert biflow data to uniflow data using the process described in Section 3.1.2, then group its data by the top-level subject in the subject hierarchy (in this implementation, \16 subnet). The mapper then sends data to a router (which, in this implementation, resides on the master node) for the reducing step.

Because the priors processing step incrementally updates internal state about calculated statistics, each individual data point must be routed to the server holding the internal state for its data. This implementation leverages the hierarchical nature

of subject priors, and maintains a reducer routing table which routes traffic based on the top-level subject hierarchy. The reducer, then, must be stateful, and it maintains state on the subject hierarchy at level 0, plus every distinct subject from lower levels of the subject hierarchy. Upon receiving requests from the router, the reducer will incrementally update its internal state using *update*. Finally, at the end of a new sliding interval, the reducer will execute *evaluate*, generating a snapshot of priors for the recent window interval. The priors are then stored in a database shared with the outlier prediction step.

Statistics & Internal State

For the majority of the calculated statistics, I use generic structures for internal state of statistics. The internal state of `prior_length` and `nan_length` for numeric, categorical, and binary types is simply a count of items and non-null items, respectively. An *evaluate* call for these statistics requires no further computation and simply returns the count. The `mean` μ also does not require internal state, as it is able to be incremented directly with a given value x based on Formula 3.15. S_n, the required internal state for `stdev` σ, requires only the mean μ (already computed in Formula 3.15) and a count of values n; it is described in Equation 3.16 [19]. We then use Equation 3.17 in the *evaluate* phase to compute σ from S_n.

$$\mu_n = \mu_{n-1} + \frac{x_n - \mu_{n-1}}{n} \tag{3.15}$$

$$S_n = S_{n-1} + (x_n - \mu_{n-1}) * (x_n - \mu_n) \tag{3.16}$$

$$\sigma = \sqrt{S_n/n} \tag{3.17}$$

The `cdf` of both categorical and binary variables is simply a count of distinct values, so the internal state holds this data as a map, and upon *update* either adds a new key initialized to 1, or increments an existing key. By knowing the `cdf`, we can calculate `category_equivalent_distance`, `category_equivalent_stdev`, and `binary_mean` as described in Section 3.1.1 during the *evaluate* phase.

The most challenging internal state to manage is `cdf` for numeric variables, which is classically computed by ordering all values in a distribution. We use the Greenwald-Khanna algorithm for space-efficient online computation of quantile summaries to store intermediate state of numeric CDF [20]. The online algorithm uses a data structure called quantile summaries, which keep track of all quantiles, up to an error of at most ϵ. The bare algorithm has a space complexity of at most $\mathcal{O}(\frac{1}{\epsilon}log(\epsilon N))$ and enables the storage and online computation of priors in this system.

3.2.2 Outlier Prediction Processing Infrastructure

Outlier predictions are nearly an embarrassingly parallel problem. Two mild but solvable requirements emerge when designing infrastructure to compute outliers:

1. **Split Biflow to Uniflows:** Each biflow data point must be split into two uniflow data points, prediction scores computed on each, then joined.

2. **Store Priors in Memory:** Optimized processing will reduce I/O costs by storing priors in memory; however, on large networks, it may not be possible to store all priors in memory.

As before, a master node will act as a load balancer to receive new data, split, and send to a slave node for prediction scoring. For this implementation, I use a cluser of Outlier Processing Machines to atomically generate a prediction score for each data point, and store the prediction in a new Outliers database. The Outlier Processing Machine uses a cache to store priors in memory, reducing I/O and increasing throughput. The processing architecture for outlier predictions is shown in Figure 3-8.
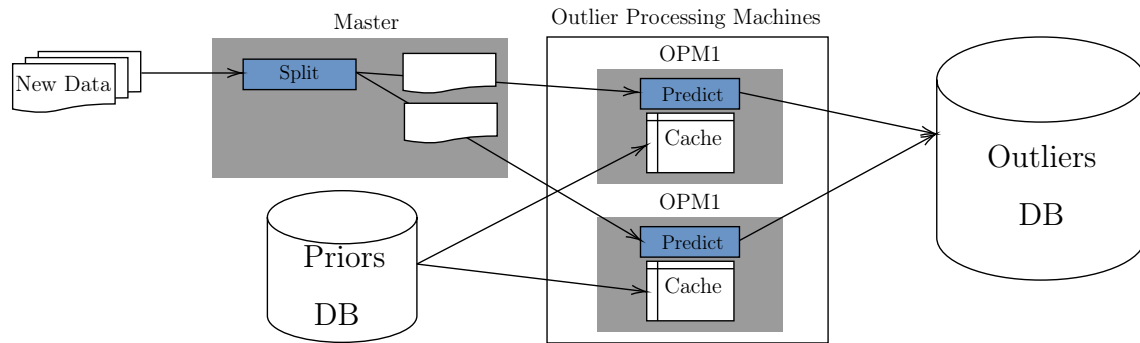
Figure 3-8: Outlier Prediction Processing Architecture

As processing needs increase, throughput can be increased by introducing routing and reducing steps to the process flow. Further performance improvements will be discussed in Section 5.2.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Results

Results of the Ensemble Outlier Detection System were displayed and analyzed using a web application developed using the Flask web framework for Python. This application, referred to as the Explorer tool, is designed to provide interpretable visualizations for every subject and object throughout a prediction's contextual tree. It also provides infrastructure for annotation of anomalies during investigations.

4.1 The Explorer Tool

4.1.1 Summary Table

The explorer tool was designed with the cybersecurity analyst as its primary user. It displays outlier prediction scores for each Netflow data point, and displays each data point in a summary table for further investigation. This summary table is shown in Figure 4-1. Generally, this table displays three types of information. The first five columns display uniquely identifiable information for the Netflow data points. This subset of fields includes source and destination IP, source and destination port, and timestamp from the flow export device. The summary table also displays a classification label, which has been configured to display either `Benign` or `Malicious` as labeled by the user. Most of these fields will be blank, as the volume of Netflow data requires that the majority of scored Netflow data points are not manually reviewed

Figure 4-1: Explorer Tool: Summary Table

by cybersecurity analysts. Finally, the summary table also includes prediction scores for each Netflow data point. In order to investigate or annotate a data point, the user can follow a link in the index column to reach the objects view. In the example in figure 4-1, the data point in index 10 has an outlier prediction of 0.77, and has been classified by a user as `Malicious`, as displayed in the figure.

4.1.2 Objects View

The objects view, shown in Figure 4-2, represents the top-level view for an individual Netflow data point. In the objects view, the user can annotate the data point as either `Benign` or `Malicious` threat level, as well as the class of threat as `Denial of Service`, `Scanning`, `Exfiltration`, or `Other`. Classifications are config-

48
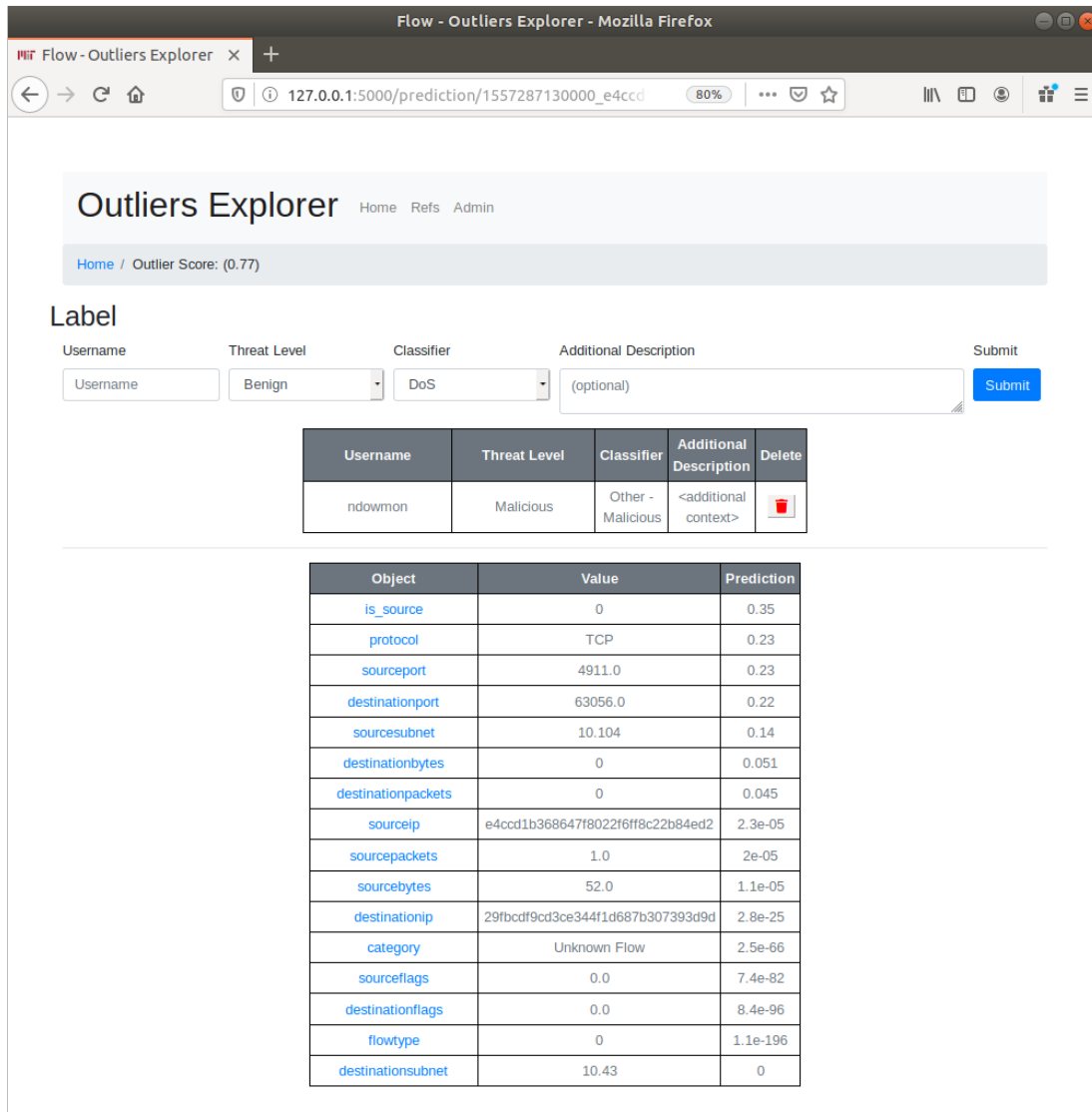
Figure 4-2: Explorer Tool: Objects View

urable for different preferences or data types. Users can also add additional classifications about their investigation, and the results of user annotations are stored in an exportable format for use in future supervised learning algorithms, discussed in Section 5.2.

Aside from annotation tools, the objects view displays each field from the Netflow data point, along with the field's value and its individual prediction score. As described in Section 3.1.3, the prediction scores in this view are combined using the recursive aggregation function (Function 3.14), maintaining a consistent overall out-

lier prediction. The objects view is ordered in descending order by outlier prediction score, and the user can further interpret the normal behavior and deviation from each field. Following the link of any object field will bring users to the subjects view.

4.1.3   Subjects View

Figure 4-3: Explorer Tool: Subjects View

The subjects view displays each independent subject which provided an outlier prediction. Depending on user configurations, there may be one or many subjects which generate outlier predictions for a given object. For example, in Figure 4-3, the value of the `protocol` field is `TCP`. In the priors generation step, priors were generated for subnet `10.43` (which is the `destinationsubnet` for this interaction), as well as the encrypted IP address beginning with `29fb...` (which is the `destinationip` address for this interaction). In this case, priors for the `sourcesubnet` (`10.104`) and the `sourceip` (encrypted, beginning with `e4cc...`, visible in Figure 4-2) have not been calculated, and thus cannot be used in outlier prediction; if they were available, this

50

subjects view would display four subjects, and an outlier prediction for each, based on the configurations of this system. Following the link of any subject brings the user into a variables view; the view differs whether the variable is categorical, binary, or numeric.

4.1.4 Categorical/Binary Variables View
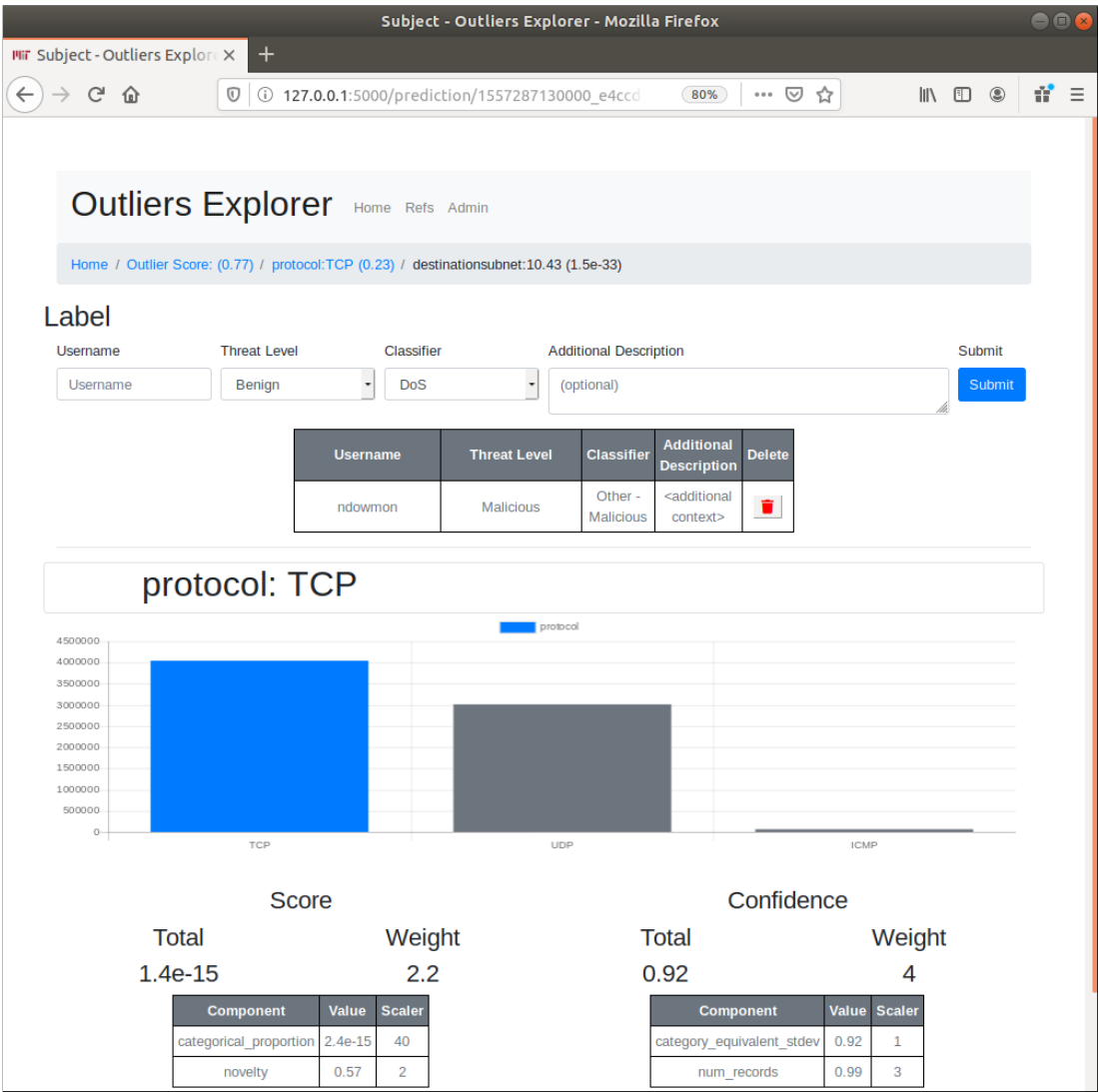
**Figure 4-4:** Explorer Tool: Categorical/Binary Variable View

The variables view presents a visual display of a variable's value against a specific subject prior. For example, in Figure 4-4, recall that the value of the `protocol` field

is `TCP`. The screen represents the distribution of all previous `protocol` values that have been encountered by the Netflow's `destinationsubnet`, `10.43` (the subject is visible near the top navigation bar).

The chart in the center of Figure 4-4 displays the distributions of protocols which have been encountered by this subnet. In the past two weeks, just over 4,000,000 Netflow data points from this subnet were of protocol `TCP` (highlighted blue, representing the value of the interaction we are investigating), over 3,000,000 data points from this subnet were of protocol `UDP`, and around 50,000 data points were of protocol `ICMP`. Recalling the functions for score and confidence described in Section 3.1.1, the extremely large number of records and relatively small distribution of categories lead to a large confidence score for the `protocol` object, from the perspective of the subject `destinationsubnet`. However, the value associated with this Netflow data point (`TCP`) is indeed the most common `protocol` value encountered by this subject, and thus, the score is quite close to zero. Looking at the distribution independently, seeing the value `TCP` gives no evidence of malicious behavior from the perspective of the `destinationsubnet`.

4.1.5   Numeric Variables View

The view for numeric variables differs from categorical and binary variables primarily in the visualization of the prior distribution. While categorical and binary variables are concerned with the count of previously seen values, numeric distributions are stored as cumulative distributions as described in Section 3.1.1. In Figure 4-5, the cumulative distribution is faint but visible as the gray line in the central chart. The `destinationip` starting with `29fb...`'s cumulative distribution has a value of 0 bytes in the 99.6$^{\text{th}}$ percentile, but a maximum value of 386 bytes. In other words, based on the behavior of the `destinationip` starting with `29fb...` in the previous two weeks, more than 99.6% of all interactions did not include it receiving any data from outside sources. In the interaction we are viewing, we see that it is indeed receiving 52 bytes of data from the source machine (represented by a blue horizontal bar in the chart). This seems to represent somewhat atypical behavior for the given machine,
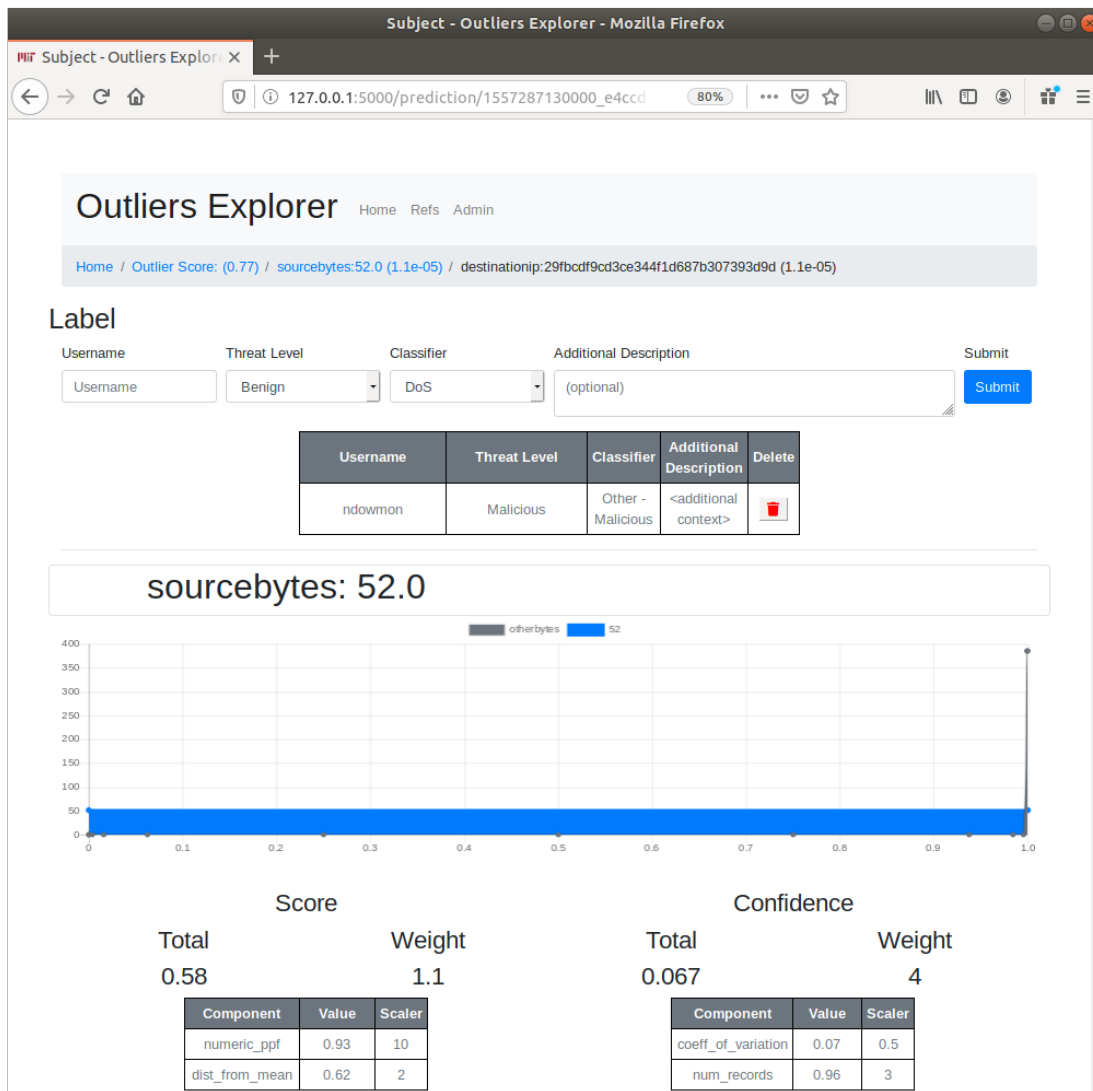
Figure 4-5: Explorer Tool: Numeric Variable View

and further investigation from a cybersecurity analyst may yield insights into this interaction.

By organizing interactions, objects, and subjects into table-based views, the Explorer tool allows analysts to quickly review and interpret newly seen behavior against baselines for each subject in the subject hierarchy. Further visualization in the variable views also lends insight into how a given behavior compares against all of the previous behavior encountered on a subject in the past. All scalers and weights are displayed at the bottom of variable views, enabling quick interpretation of major

influences on outlier predictions, which are combined as defined in Section 3.1.3 to generate the outlier prediction, based on all available subject context.

4.2   Score Distributions and Performance

The Ensemble Outlier Detection System was designed with flexibility in mind, as evidenced by the addition of scalers to both score and confidence components, as well as score weight ($ScoreWt$) and confidence weight ($ConfWt$) values. The wide variety of coefficients in the outlier prediction process enables various optimization strategies to ensure that the highest outlier scores are nearly evenly distributed across different data fields. For example, given a set of outlier predictions, if the scores for the top 100 outlier predictions are uniformly and independently driven by high outlier predictions for the data point's `protocol` field, the scaling factors are probably weighing the `protocol` field unevenly compared to other fields. With the current dataset, I chose to optimize the coefficients in the following way:

1. Generate outlier predictions for the first 100,000 data points

2. Set the scalers of every score component and confidence component, as well as the confidence weight $ConfWt$, to static values

3. Optimize the score weight $ScoreWt$ on each nondirectional and directional field, such that 0.1% of all independent outlier predictions for that variable have a value greater than 0.1

Priors were generated based on one week of data using the subnet selected in Section 2.2. Outlier predictions were then generated for the subsequent 100,000 data points, representing an 5 hour time interval. The scalers and confidence weight $ConfWt$ were set to the static values shown in Table 4.1. Static values were chosen based on visual interpretation of score component and confidence component graphs, shown in Figures 3-2 and 3-3.

Holding the values in Table 4.1 constant, the $ScoreWt$ was then optimized such that approximately 0.1% of outlier predictions result in scores above 0.1. Because

54

Table 4.1: Static Scalers and Confidence Weight

Scaler	Value
$Score_{numeric\_ppf}$	10
$Score_{dist\_from\_mean}$	2
$Score_{categorical\_proportion}$	40
$Score_{novelty}$	2
$Conf_{num\_records}$	3
$Conf_{coef\_of\_var}$	0.5
$Conf_{category\_equivalent\_stdev}$	1
$Conf_{binary\_mean}$	2
$ConfWt$	4

outlier predictions were generated on static data, some fields could not be optimized to near 0.1%. The results of the optimization, and the associated $ScoreWt$ coefficients, are shown in Table 4.2.

Table 4.2: Score Weights for Optimized Coefficients

Field	ScoreWt Coefficient	Outlier Predictions > 0.1
IP	1.5356	0.018%
Port	1.8826	0.093%
Subnet	2.842	0.040%
Flags	2.32	0.109%
Bytes	1.1299	0.095%
Packets	1.1823	0.100%
Protocol	2.2	0.102%
Category	2.84	0.107%
Flowtype	2.2	0.107%
Is Source	0.8703	0.083%

After optimizing each individual field, the combined outlier prediction scores were re-computed using the outlier prediction process. The results of the 100,000 outlier predictions are shown in Figure 4-6. Note that the Y-axis is displayed using a logarithmic scale.
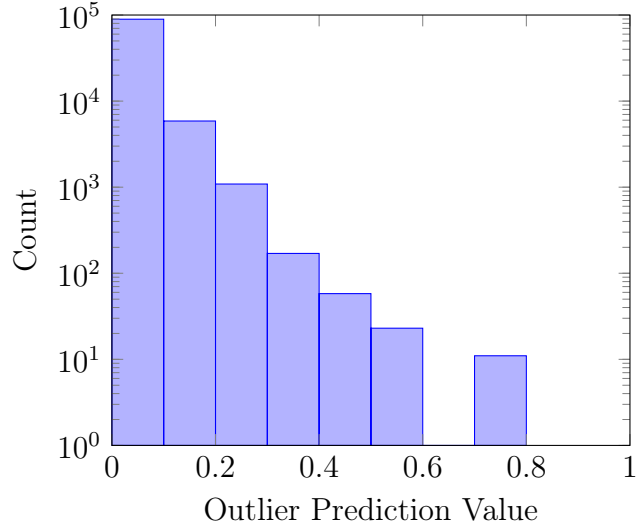
Figure 4-6: Histogram of Outlier Predictions for 100,000 Netflow Data Points

The histogram in Figure 4-6 shows roughly exponential decay as outlier prediction bins increase from $\{0.0 \to 0.1\}$ to $\{0.7 \to 0.8\}$. In the 100,000 data points for which outlier prediction scores were calculated, none had predictions above 0.765. With more data points (for example, from an entire day), we may expect to see some strong outliers emerge in the 0.9-1.0 range. However, the 100,000 data points used in this analysis were from the hours of 12:00 AM - 05:00 AM, which may be a low-volume window of time on the network.

Performance of the system was tested using a client machine running Ubuntu 18.04.3. The machine uses an Intel i7-8750H CPU and has 32 GB of memory and 512 GB SATA3 Solid State Drive. The priors generation process was executed using 1 week of data from a single subnet, including 7,103,882 Netflow data points with the 17 fields referenced in Section 2.2. The generation of priors took 1 hour and 48 minutes. The priors generation resulted in 16,993 unique priors, totaling 69.4 MB of data.

The outliers generation process was executed using 100,000 Netflow data points, including approximately 5 hours of data from the same subnet referenced above. The generation of outlier predictions completed in 24 minutes, totaling 314.5 MB of data. Outlier predictions seem to take much longer than priors due to the fact that priors are currently stored on disk, and loaded into memory as baseline references to compute

outlier predictions. A number of optimizations come to mind to reduce processing time on local machines, most notably using an in-memory database or file system to store priors during the outlier generation process.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Discussion

The methods introduced in the Ensemble Outlier Detection System are able to reduce space complexity and manual configurations, and thereby increase scalability and resiliency to non-stationary data. The system generates outlier predictions on each field in a data point independently, and combining those predictions to generate an outlier prediction for the data point. This process relies on the assumption that Netflow data points exhibiting anomalous behavior will most likely display uncharacteristic behavior in multiple independent fields.

Therefore, the Ensemble Outlier Detection System can be broadly critiqued by answering the following question: Are outlier behaviors characterized by multiple unexpected field values? To answer this question, I present a modified correlation matrix, shown in Figure 5-1.

The rows of the matrix were created using fields with an outlier prediction score greater than 0.1. For example, when computing the values for row 1, the entire 100,000 point Netflow dataset was filtered by only those data points whose `sourceip` outlier prediction scores were greater than 0.1. Columns of the modified correlation matrix represent the average outlier prediction score of each other field in the data point. For all data points whose `sourceip` was greater than 0.1, the outlier prediction score of the `sourcesubnet` field was, on average, 0.08, and the outlier prediction score of the `destinationport` was, on average, 0.05. The remaining fields generally did not exhibit outlier behavior in this case. This shows that when the `sourceip` field

Figure 5-1: Modified Correlation Matrix, Fields with Predictions > 0.1

was an outlier, oftentimes the `sourcesubnet` and/or the `destinationport` was also predicted as an outlier.

The modified correlation matrix is not a square matrix, but it does show many reciprocal relationships; for example, when the Source Bytes field is predicted as an outlier, so too is the Source Packets field, and vice versa - but other fields show little to no correlation. The same is true of Destination Bytes and Destination Packets. This conclusion makes sense, given that packets and bytes are two fields with a strong correlation to begin with.

Other interesting patterns emerge from the modified correlation matrix. For example, when the Source Port has high outlier prediction scores, so too do Source Subnet, Destination Port, Destination Bytes, Destination Packets, Protocol, and Is Source. However, when the Destination Port has high outlier prediction scores, virtually the same fields have high outlier prediction scores; in both cases, Destination Bytes and Destination Packets are predicted as anomalous, but neither case has any

correlation with high outlier prediction scores in Source Bytes or Source Packets. The cause for this discrepancy is uncertain, but it may suggest that when machines are operating on different ports, they are receiving more data from the destination than typical (this may represent malicious behavior, or possibly something more benign, like unusual but valid VPN access of a client machine).

The highest value on the modified correlation matrix is 0.14, suggesting that when the Source Port field has a high outlier prediction score, the Destination Port is almost always predicted as anomalous as well. Other high correlations include Source Bytes/Source Packets and Destination Bytes/Destination Packets (for reasons described above), Destination IP/Destination Subnet (suggesting that machines which generally interact with specific subnets also interact with specific machines), and Source Flags/Destination Flags (as TCP flags are correlated in the forward and reverse directions).

Of all the 17 fields within the Netflow data points, Source Subnet and Flow Type show little correlation with other fields. Otherwise, this matrix suggests strong evidence that when one field is exhibiting anomalous behavior, there is often at least one other field exhibiting anomalous behavior. This shows, at least preliminarily, that generating outlier prediction scores on each independent field enables scalability and resiliency to non-stationary data, while still identifying high-risk anomalies in the network.

5.1 Related Work

The Ensemble Outlier Detection System is a novel work in anomaly detection research, and it differentiates from other anomaly detection systems primarily by its scalability, since it treats fields within Netflow data points independently of one another, and by its resiliency to non-stationary data, since it does not require manual configurations to re-establish a network baseline. However, two papers are notable in their relation to this work.

In [21], authors use machine learning models to generate a collection of anomalies,

and in a later step, filter these anomalies to alert only on relevant anomalies for the context. The system is extensible and allows for additional models to be integrated into the system. The system is both generic and scalable, but using default models, while the Ensemble Outlier Detection System introduces newly defined prediction algorithms for numeric, categorical, and binary variable types.

In [22], authors distinguish between *normal* traffic, *anomalies*, and *outliers* using a confidence algorithm and additional context in order to design a generalizable anomaly detection system which can work across different domains. The framework differs from the Ensemble Outlier Detection System primarily in its lack of interpretability as a principal design component.

5.2   Future Work

The Ensemble Outlier Detection System could be enhanced in the future by adding interpretable score and confidence components for additional data types. One notable example is the datetime data type; in many contexts, including network traffic data, the time at which an event occurs can indicate anomalous behavior. For example, given a client machine which only operates between the hours of 9:00 AM - 5:00 PM on weekdays, an interaction at 2:30 AM on a Monday may indicate infiltration or some other malicious behavior.

The system has implemented infrastructure to annotate data points and export annotations in bulk as training data. Currently, the bottleneck in developing machine learning algorithms for classification is primarily a scarcity of annotated data. With annotated data, the system could be enhanced to predict classifications of anomalies, further optimizing workloads of data points which require manual review from cybersecurity analysts.

With the collection of scalers in the source and confidence components, plus score and confidence weights, the possibilities for optimizing model parameters are currently an under-constrained problem. Further investigation into how best to use the system's flexibility in parameter tuning could yield improved results from the anomaly

detection system.

Real-time processing infrastructure for the system is designed for test, but the web application for interpretation and annotation is only designed for use with static datasets from the non-streaming processes. In order to make the system fully functional in production environments, it requires additional integrations and long-term support.

The main focus of this project was to design a functional tool for use in the sponsor's CDC that could utilize new technology to identify anomalies in their network traffic data. Therefore, all tests and metrics were produced using data from the client's network. However, in order to benchmark this system against other anomaly detection techniques, results should be generated using open and annotated network traffic datasets, such as those described in [23].

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6

Conclusion

In today's world, every company is a technology company. Enterprise businesses are capturing and hosting more data than ever before, and have massive liabilities if malicious actors are able to infiltrate an organization's computer network. Businesses are preparing for persistent attacks by using up-to-date security software capable of identifying and triaging known attack signatures, stopping malicious actors in their tracks. However, when it comes to the unknown attack signatures of new (zero-day) attacks, the best defense is knowing the expected behavior of the computer network, and identifying deviations from that behavior using anomaly detection techniques. These techniques can be effective, but are often difficult to scale, especially when looking at computer networks of hundreds of thousands of machines.

In order to meet the needs of today's cyber defense teams, I pursue the development of a system with the following six requirements: Resiliency, Scalability, Atomicity, Intrepretability, Supervised Learning Annotations, and Flexibility. By meeting these requirements, the anomaly detection system can identify previously unknown attacks and become a valuable tool to enterprise cybersecurity teams.

To achieve these requirements, I build on existing anomaly detection research and concepts to design a new framework for detecting anomalies in multidimensional network traffic data. I present the Ensemble Outlier Detection System, a two-step process for creating a baseline of network behavior, calculating outlier prediction scores, and presenting interpretations of those results. This system borrows concepts

from Denning [14] to describe subject-object relationships. It deviates from existing systems by assuming zero covariance between fields, treating them as independent. This tactic ensures a space complexity of approximately $\mathcal{O}(n)$, enabling a scalable framework even on large computer networks.

I also describe the processing infrastructure used to ensure real-time online processing of outlier predictions. I borrow the concept of windowed operations from Apache Spark Streaming [18], as well as MapReduce architecture, to ensure the ability of streaming outlier predictions.

I develop and introduce the Explorer tool, a Flask web application designed with the cybersecurity analyst as its primary user. The Explorer tool enables interpretation of results from the Ensemble Outlier Detection System, displaying hierarchical views of each field within a Netflow data point. It also retrieves and displays the priors that each data point is compared against. The Explorer tool presents data visualizations for numeric, categorical, and binary data types, and also hosts annotations from cybersecurity analysts, which can be exported for use in supervised learning methods, described in future work.

The system was tuned based on data provided by the research sponsor. The Ensemble Outlier Detection System was executed on a local machine, and was able to generate priors from 2 weeks worth of data in less than two hours, and generate 5 hours worth of outlier scores in less than 30 minutes. The performance suggests that the test subnet of approximately 300 data points per minute would be able to maintain real-time outlier predictions, and by moving the production system to the online processing infrastructure, much larger networks could be accommodated.

A modified correlation matrix computed against the anomalous fields suggests that when one field in the Netflow data point is scored as anomalous, there is often at least one other field which is also anomalous. The inference that anomalous data points often present with at least two anomalous fields is an indication that treating fields as independent of one another and combining scores is a valid method to capture anomalies on a computer network. This method is critical for maintaining scalability to meet the processing requirements of very large systems.

The system uses a completely novel set of mathematics to calculate priors and outlier predictions for numeric, categorical, and binary data types in Netflow data. Future work includes benchmarking the system against other anomaly detection techniques using open network traffic datasets. Beyond network traffic, this generic framework can be applied in the future to finance, retail, healthcare, government, and many other outlier detection use cases.

Glossary

Attack Surface The sum of the different points (the "attack vectors") where an unauthorized user (the "attacker") can try to enter data to or extract data from an environment. Keeping the attack surface as small as possible is a basic security measure.. 1, 7

Exfiltration Data theft which occurs when malware and/or a malicious actor carries out an unauthorized data transfer from a computer. 9

Directional Key Field A Directional Key Field is a single field in a Flow Key as defined in the IPFIX Protocol document [RFC5101] that is specifically associated with a single endpoint of the Flow. sourceIPv4Address and destinationTransportPort are example Directional Key Fields. 10, 11, 36

Nondirectional Key Field A Non-directional Key Field is a single field within a Flow Key as defined in the IPFIX Protocol document [RFC5101] that is not specifically associated with either endpoint of the Flow. protocolIdentifier is an example Non-directional Key Field. 10, 11, 36

Uniflow A Unidirectional Flow (Uniflow) is a Flow as defined in the IPFIX Protocol document [RFC5101], restricted such that the Flow is composed only of packets sent from a single endpoint to another single endpoint. 10, 36, 44

Biflow A Bidirectional Flow (Biflow) is a Flow as defined in the IPFIX Protocol document [RFC5101], composed of packets sent in both directions between two endpoints. A Biflow is composed from two Uniflows such that:

1. the value of each Non-directional Key Field of each Uniflow is identical to its counterpart in the other, and

2. the value of each Directional Key Field of each Uniflow is identical to its reverse direction counterpart in the other

A Biflow contains two non-key fields for each value it represents associated with a single direction or endpoint: one for the forward direction and one for the reverse direction, as defined below. 10, 33, 36, 44

Biflow Source The Biflow Source is the endpoint identified by the source Directional Key Fields in the Biflow. 10, 36, 37

Biflow Destination The Biflow Destination is the endpoint identified by the destination Directional Key Fields in the Biflow. 10, 36, 37

Forward Direction The direction of a Biflow composed of packets sent by the Biflow Source. Values associated with the forward direction of a Biflow are represented using normal Information Elements. In other words, a Uniflow may be defined as a Biflow having only a forward direction. 10, 36

Reverse Direction The direction of a Biflow composed of packets sent by the Biflow Destination. Values associated with the reverse direction of a Biflow are represented using Reverse Information Elements, as defined below. 10, 36

Reverse Information Element An Information Element defined as corresponding to a normal (or forward) Information Element, but associated with the reverse direction of a Biflow. 10, 36

Flow Export Device A network device used for packet observation (capturing packets from the line), metering (aggregation of packets into flows), and export to Flow Collectors. Flow export devices include routers and network switches.. 11, 47

Flow Collector A network device used for storage and pre-processing of data sent by Flow Export Devices. 11

Prior Past behavior of a given subject used for comparisons. 16, 27, 30

Bibliography

[1]   Dell Inc. *NIST Cybersecurity Framework Overview*. Mar. 2019. URL: `https://infocus.dellemc.com/michael_dulavitz/strengthen-security-of-your-data-center-with-the-nist-cybersecurity-framework/` (visited on 11/16/2019).

[2]   Manasa Chalasani [Awake Security]. *Getting into a SOC Analyst's Head*. Awake Security. 2018. URL: `https://awakesecurity.com/blog/getting-soc-analysts-head/` (visited on 11/16/2019).

[3]   *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1.* Tech. rep. Apr. 2018. DOI: `10.6028/nist.cswp.04162018`. URL: `https://doi.org/10.6028/nist.cswp.04162018`.

[4]   Michael Howard, Jon Pincus, and Jeannette M. Wing. "Measuring Relative Attack Surfaces". In: *Computer Security in the 21st Century*. Springer-Verlag, pp. 109–137. DOI: `10.1007/0-387-24006-3_8`. URL: `https://doi.org/10.1007/0-387-24006-3_8`.

[5]   Antonio Regalado. "Business Adapts to a New Style of Computer". In: *MIT Technology Review* (May 2014). URL: `https://www.technologyreview.com/s/527356/business-adapts-to-a-new-style-of-computer/`.

[6]   Hanan Hindy et al. "A Taxonomy of Malicious Traffic for Intrusion Detection Systems". In: *CoRR* abs/1806.03516 (2018). arXiv: `1806.03516`. URL: `http://arxiv.org/abs/1806.03516`.

[7] Hamzah Al Najada, Imad Mahgoub, and Imran Mohammed. "Cyber Intrusion Prediction and Taxonomy System Using Deep Learning And Distributed Big Data Processing". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2018, Bangalore, India, November 18-21, 2018*. 2018, pp. 631–638. DOI: 10.1109/SSCI.2018.8628685. URL: https://doi.org/10.1109/SSCI.2018.8628685.

[8] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018*. 2018, pp. 108–116. DOI: 10.5220/0006639801080116. URL: https://doi.org/10.5220/0006639801080116.

[9] Eric Ahlm et al. *Predicts 2017: Network and Gateway Security*. Tech. rep. G00317597. Gartner Research, Dec. 2016. URL: https://www.gartner.com/en/documents/3542117.

[10] B. Trammell and E. Boschi. *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)*. Tech. rep. Jan. 2008. DOI: 10.17487/rfc5103. URL: https://doi.org/10.17487/rfc5103.

[11] *Cisco IOS NetFlow Version 9 Flow-Record Format*. Tech. rep. C11-395693-01. May 2011. URL: https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.pdf.

[12] K A Scarfone and P M Mell. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Tech. rep. 2007. DOI: 10.6028/nist.sp.800-94. URL: https://doi.org/10.6028/nist.sp.800-94.

[13] Danfeng (Daphne) Yao et al. "Anomaly Detection as a Service: Challenges, Advances, and Opportunities". In: *Synthesis Lectures on Information Security, Privacy, and Trust* 9.3 (Oct. 2017), pp. 1–173. DOI: 10.2200/s00800ed1v01y201709spt022. URL: https://doi.org/10.2200/s00800ed1v01y201709spt022.

[14] D.E. Denning. "An Intrusion-Detection Model". In: *IEEE Transactions on Software Engineering* SE-13.2 (Feb. 1987), pp. 222–232. DOI: `10.1109/tse.1987.232894`. URL: `https://doi.org/10.1109/tse.1987.232894`.

[15] Ahmed AlEroud and George Karabatis. "Methods and techniques to identify security incidents using domain knowledge and contextual information". In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, May 2017. DOI: `10.23919/inm.2017.7987435`. URL: `https://doi.org/10.23919/inm.2017.7987435`.

[16] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection". In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–58. DOI: `10.1145/1541880.1541882`. URL: `https://doi.org/10.1145/1541880.1541882`.

[17] Edward Yu and Parth Parekh. "A Bayesian Ensemble for Unsupervised Anomaly Detection". In: *arXiv e-prints*, arXiv:1610.07677 (Oct. 2016), arXiv:1610.07677. arXiv: `1610.07677 [stat.ML]`.

[18] *Apache Spark*. 2014. URL: `http://spark.apache.org/`.

[19] B. P. Welford. "Note on a Method for Calculating Corrected Sums of Squares and Products". In: *Technometrics* 4.3 (Aug. 1962), pp. 419–420. DOI: `10.1080/00401706.1962.10490022`. URL: `https://doi.org/10.1080/00401706.1962.10490022`.

[20] Michael Greenwald and Sanjeev Khanna. "Space-efficient Online Computation of Quantile Summaries". In: *SIGMOD Rec.* 30.2 (May 2001), pp. 58–66. ISSN: 0163-5808. DOI: `10.1145/376284.375670`. URL: `http://doi.acm.org/10.1145/376284.375670`.

[21] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. "Generic and Scalable Framework for Automated Time-series Anomaly Detection". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*. ACM Press, 2015. DOI: `10.1145/2783258.2788611`. URL: `https://doi.org/10.1145/2783258.2788611`.

[22] Evana Gizzi et al. "A Generalized Framework for Detecting Anomalies in Real-Time using Contextual Information". In: *Proceedings of the Tenth International Workshop Modelling and Reasoning in Context*. ACM Press, 2018.

[23] Markus Ring et al. "A survey of network-based intrusion detection data sets". In: *Computers & Security* 86 (Sept. 2019), pp. 147–167. DOI: `10.1016/j.cose.2019.06.005`. URL: `https://doi.org/10.1016/j.cose.2019.06.005`.