Utilizing Threat Partitioning for More Practical Network Anomaly Detection

Brian Ricks
The University of Texas at Dallas
Computer Science Department
Richardson, TX, USA
absolutefunk@utdallas.edu

Patrick Tague
Carnegie Mellon University
Information Networking Institute
Moffett Field, CA, USA
tague@cmu.edu

Bhavani Thuraisingham
The University of Texas at Dallas
Computer Science Department
Richardson, TX, USA
bxt043000@utdallas.edu

Sriraam Natarajan
The University of Texas at Dallas
Computer Science Department
Richardson, TX, USA
Sriraam.Natarajan@utdallas.edu

ABSTRACT

Anomaly-based network intrusion detection would appear on the surface to be ideal for detection of zero-day network threats. Yet in practice, their often unacceptably high false positive rates keep them on the sideline in favor of signature-based methods, which typically detect known threats. We argue that an anomaly-based network intrusion detection system should not only be specialized to a specific class of related threats, but characteristics of the threat class itself should be utilized when designing both the detection system and structuring the network data to use with the system. To this end, we take two common network threat classes, DDoS-as-a-Smokescreen (DaaSS) and SYN flood, and analyze their characteristics for structure that we can use to specialize anomaly detection. We partition these threat classes into known behavior and unknown behavior, leaving the latter open-ended. Through experimentation on multiple datasets, we show that our proposed detection system based on this threat partitioning approach is capable of detecting DaaSS attacks and zero-day SYN flood variants with very low false positive rates, even in the face of concept drift, and can do so without having to collect large amounts of benign network traffic for training.

CCS CONCEPTS

• **Security and privacy** → **Denial-of-service attacks**; **Intrusion detection systems**; • **Networks** → *Network monitoring*.

KEYWORDS

network intrusion detection, anomaly detection, threat partitioning, DDoS, DaaSS, SYN flood, netflow

ACM Reference Format:
Brian Ricks, Patrick Tague, Bhavani Thuraisingham, and Sriraam Natarajan. 2024. Utilizing Threat Partitioning for More Practical Network Anomaly Detection. In *Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT 2024), May 15–17, 2024, San Antonio, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3649158.3657046

This work is licensed under a Creative Commons Attribution International 4.0 License.

SACMAT 2024, May 15–17, 2024, San Antonio, TX, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0491-8/24/05
https://doi.org/10.1145/3649158.3657046

1 INTRODUCTION

Network intrusion detection systems (IDS) have grown from modest roots to an entire industry where network security vendors provide intrusion detection as a service. These vendors deploy monitors on their client networks which compare network traffic against known threats, raising alarms if any traffic matches a threat. Known threats are typically represented as signatures, leading to *signature-based* IDS capable of threat detection with a zero false alarm rate, but which are blind to unknown, or zero-day, threats. To protect their clients from zero-day threats, security vendors attempt to discover them in the wild before an attacker does.

On the surface it may seem appropriate to utilize *anomaly-based* network intrusion detection systems to detect zero-day threats. These IDS do not encompass knowledge of known threats, instead considering as a potential threat any network traffic whose behavior deviates from what the IDS considers 'normal', or benign. If an anomaly is detected, the IDS will not know if the anomaly is actually a zero-day threat or benign behavior which happened to deviate.

Due in large part to this unacceptably high false alarm rate, anomaly-based network intrusion detection systems are rarely used in practice [27]. Considering that false alarms may bring about a loss of trust in the IDS, and also considering that large security vendors can often push out new signatures very quickly after zero-day threat discovery, there needs to be a strong case to use anomaly-based IDS in the real-world.

We argue that real-world advancement of anomaly-based IDS is hindered because they are asked to do too much in too broad of a setting. Detecting threats in this setting is often like searching for a needle in pile of haystacks. The IDS knows what hay looks like, but not the needle, searching blindly through the pile, finding anything that does not resemble hay. The haystack pile may also change over time, perhaps with new types of hay introduced which the IDS does not recognize. A signature-based IDS, on the other hand, does not need to know what hay is, as it knows specific needles.

The haystack analogy brings to light key issues inherent with anomaly-based methods which directly contribute to a high false alarm rate, namely a lack of scoping/direction in the threat search, and inability to adapt as the benign traffic landscape changes over time. Additionally, challenges in data procurement necessary for anomaly detection model training and testing present another key issue that may eliminate anomaly-based IDS from consideration.

We draw inspiration from the seminal work of Sommer and Paxson [27], in which limitations of anomaly-based network intrusion detection systems are discussed. Sommer and Paxson argued that for anomaly-based IDS to be practical, the IDS needs to have some idea of the threats it is supposed to detect. Clearly, if the IDS knows exactly the threats it should search for, then we can use signature-based methods, but if we can incorporate partial knowledge of the threats we want the IDS to detect, then we can perhaps guide the IDS in the search.

To incorporate partial threat knowledge, we introduce a novel approach called *threat partitioning*, which scopes the threat search both in terms of what specific threats we search for and the time periods in which we search for them. Threats are grouped into classes based on known behavior common to them, and for a threat class that we would like to detect, we only consider its common behavior as known, leaving the rest as an anomaly detection task. This constrains anomaly detection to only the time periods in which a threat's known behavior is observed.

Procurement of benign traffic required to train anomaly-based IDS is often a challenge [22, 27]. By utilizing threat partitioning, we greatly reduce the amount of benign traffic which we need to collect to traffic occurring only in the presence of known threat behavior. In other words, the IDS only needs to know what hay looks like in the presence of a partial needle.

Lastly, hyper-parameters used for anomaly-based IDS sensitivity tuning may be unintuitive to set. Unfortunately, we may not know how well the IDS can detect threats with a specific tuning until the network it is monitoring is attacked. By utilizing network traffic data which captures concept drift, or variations in behavior over time, we automatically set sensitivity hyper-parameters in a manner conducive for maintaining IDS performance in the face of ever-changing network traffic patterns.

Our key contributions are as follows:

- We address the real-world practicality of anomaly-based IDS by scoping the threat search using an approach we call threat partitioning.
- We design an IDS incorporating threat partitioning which requires minimal benign network traffic to train, and no sensitivity hyper-parameters to tune.
- We show through experimentation utilizing two real-world threat classes that our proposed IDS can detect a wide range of zero-day threats with few false alarms, and can do so in the presence of concept drift.

The remaining sections are structured as follows. First we discuss preliminaries and related work in Section 2, including an introduction of the DDoS-as-a-smokescreen and SYN flood threat classes used as case studies. Section 3 introduces our approach pipeline, starting with an introduction to threat partitioning in Section 3.1, structuring of network packet data in Section 3.2, and threat detection models in Section 3.3. Section 4 dives into our experiments, including a description of the datasets used and experimental methodology, with results given in Section 4.3. We then discuss limitations of our approach in Section 5, and conclude our work in Section 6.

2 PRELIMINARIES

In the following sections we discuss relevant background, namely anomaly-based network intrusion detection methods, a network data format useful for model training and testing, and two threat classes we will use for experimentation.

2.1 Anomaly-Based Network Intrusion Detection

Network intrusion detection systems have been discussed since at least the 1980s [5], becoming commonplace in the early Internet around the turn of the century [25]. One of the first IDS capable of anomaly detection that saw widespread adoption was Bro [20], now called Zeek. Zeek uses a set of rules to define benign network behavior, taking deviations from the rules to be anomalous. As machine learning methods gained in popularity, anomaly-based IDS shifted to paradigms in which the representation of benign network behavior was learned from network data [29]. For the remainder of this work, we will consider an 'anomaly-based IDS' to comprise a model or ensemble of models learned from data, using machine learning methods.

Anomaly-based IDS utilizing machine learning methods typically fall into the category of one-class classifiers (OCC), in which a single class representation is learned from benign traffic data. Once trained, the model is capable of binary classification utilizing a distance metric to determine if an example is benign or anomalous [15].

Our prior work includes a generalization of OCC for multi-class classification using hard-clustering [24]. Here, a k-Prototypes model [11] is learned from multi-class network traffic data, with the clusters bounded by a radius to induce a space for anomalous classification. Masud et. al [18] also utilized a radius as a distance metric in a k-Means model used for novelty detection.

Soft-clustering models supporting anomaly detection typically use likelihood as a distance metric for anomalous classification. Application domains where this type of approach has been used for anomaly detection includes rocket engine diagnostics [17] and web mining [16].

As previously discussed, a major limitation of anomaly-based network intrusion detection systems is their false alarm rate. The key issues we address in this work, among others, were brought to light in 2010 by Sommer and Paxson [27], who also pointed out the lack of industry adoption for such systems.

2.2 Network Flow Data

Traffic collected from networks are typically raw packet captures (PCAP), which lack inter-packet structure needed for model training. To provide such structure, we utilize a data format which represents packet data as network flows, called netflows [10]. A netflow (or simply *flow*) represents a sequence of packets to and from a pair of network nodes, defined as a 5-tuple: source IP address, destination IP address, source port, destination port, and protocol (TCP for example). Netflows additionally comprise attributes representing aggregates for characteristics such as the number of packets that comprise a flow.

A flow begins when a packet with a unique 5-tuple arrives at a network node, and ends when a timeout criteria is met, either due to traffic inactivity or due to a maximum duration limit being reached. The duration of a netflow is the time interval between the arrival times of the first and last packet of the flow. The network node which receives the first packet in a flow becomes the destination for the flow. When a flow ends, subsequent packet arrivals with the same 5-tuple will form a new flow. [1]

The resulting format is a propositional flat file in which each row is a netflow, and each column is an attribute value. Netflow datasets can be used directly for machine learning by mapping each flow as an example, and each attribute as a feature [9]. Domains in which netflow datasets have been used to train models include bot detection on social networks [8], and bot command-and-control (c&c) characterization [7].

2.3 Threat Class: DDoS-as-a-Smokescreen

A common threat class frequently observed in the wild is DDoS-as-a-Smokescreen (DaaSS), an ensemble of two threats that occur together. The first threat is a Distributed Denial-of-Service (DDoS), designed to obfuscate a second threat, which may be a zero-day [24]. The earliest known DaaSS attack occurred in 2011 on the Sony PlayStation Network, resulting in the exfiltration of personal data from approximately 77 million users [30]. This data exfiltration was not discovered until after the DDoS ceased, due in large part to mitigation focused on the DDoS itself.

Since the Sony PlayStation Network attack, DaaSS attacks have become more prevalent. According to the 2016 Kaspersky Lab Corporate IT Security Risks survey, over half of the respondents (56%) believed that DDoS attacks they had been subjected to were used as a smokescreen [14]. StormWall's H1 2023 DDoS attack report noted that DaaSS attacks rose 26% from the previous year [28]. Recent DaaSS attacks include the FlexBooker data breach [12], which specifically involved the exfiltration of personal data.

One reason why DaaSS attacks are particularly effective is they exploit limited IT resources who may be aware of the smokescreen but have little choice in fighting the DDoS. They are also relatively cheap to launch using third parties [13].

2.4 Threat Class: SYN Flood

A SYN flood is class of denial-of-service threats targeting the TCP protocol's three-way handshake connection initialization process. In a client-server paradigm, the three-way handshake starts with the client sending a SYN packet to the server. The server responds with a SYN+ACK packet, and upon receipt of this SYN+ACK packet, the client replies with an ACK packet back to the server.

This three-way handshake process is exploited during a SYN flood, in which many SYN packets are sent to a server by one or more clients. When the server replies with SYN+ACK packets, the clients do not respond with ACKs, resulting in increased resource usage at the server to keep track of the many potential connections. Eventually the server may exhaust resources and be unable to process new connection attempts. SYN floods can be executed using a single client, spoofing the source address of each SYN packet sent to make it seem as though the many connection attempts are coming from different clients.

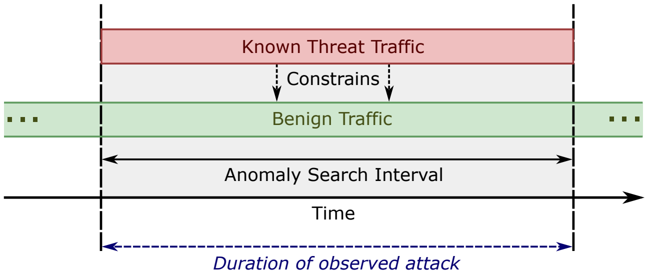

Figure 1: Illustration of threat partitioning. Anomaly detection is reduced to the intervals in which the known threat traffic is present. The behavioral characteristics of such traffic may constrain benign traffic that is simultaneously present.

SYN flood attacks represent one of the oldest denial-of-service, and are still common today. In 2020, a customer of security vendor Imperva was hit with a zero-day SYN flood variant. According to Imperva [2], the SYN packets comprising the flood contained payloads, which is not recognized by RFC-793, the document describing SYN packet structure [21]. Furthermore, this same attack comprised spoofed ACK packets, perhaps in an attempt to fake completion of the three-way handshake.

3 APPROACH PIPELINE

We formally define threat partitioning in Section 3.1, including requirements we deem are necessary for a threat to be partitioned. We then discuss network traffic structuring in Section 3.2, and the detection models which comprise our proposed IDS in Section 3.3.

3.1 Threat Partitioning

Rather than treat a network attack as an atomic, unknown unit, as is the case with general anomaly detection, we propose to partition the attack into two components: the known behavior, and unknown behavior (anomalies). This involves utilizing a model specialized to detect the traffic which we expect to observe, and if such traffic is detected, only then attempt to detect anomalies. Therefore, false positives would be specific to either known threat traffic or anomalous traffic. We expect our IDS to minimize known threat traffic false positives due to such threat behavior often derived from bots or malware, leading to more determinism in behavioral patterns.

In addition to this anomaly detection interval constraint, we should consider how benign traffic may be affected by known threat traffic, for example through bandwidth or resource exhaustion. Consider a local-area network comprising one router that connects it to the Internet, providing a single point for all network traffic to flow. Suppose the local-area network is attacked with a high bandwidth inbound DDoS. We would expect benign traffic to be affected due to inbound bandwidth exhaustion, so anything odd, such as a large outflow of traffic, may point to a DaaSS with data exfiltration rather than a DDoS in isolation.

[1] For TCP flows, the arrival of a SYN packet triggers a new flow.

By partitioning known threat behavior from the unknown, we are also simplifying the anomalies to a subset of the overall threat behavior. For example, to detect a zero-day SYN flood variant, we would not need to consider known SYN flood behavior when searching for anomalies, as this would already be detected. With general anomaly detection, the SYN flood variant in its entirety would need to be detected as anomalous, even though some of its behavior is already known.

Figure 1 illustrates characteristics of threat partitioning. The known threat traffic constraint on benign traffic is present only during the interval in which known threat traffic itself is present. Thus, we can consider benign traffic present outside this interval as being independent of the benign traffic present inside. In addition to reducing the false positive rate by constraining the anomaly search interval, this independence assumption on benign traffic implies that we do not need to collect benign traffic in the absence of any known threat behavior, which we expect to be the vast majority of the time. This is a huge advantage, as in general collecting representative benign traffic is a difficult problem [26, 27].

Without threat partitioning, if we wanted to detect data exfiltration using traditional anomaly detection methods, we could train an OCC model for this detection task. In addition to the enormous amount of benign traffic we would need to collect for model training, it may be difficult in the general case to separate a malignant data exfiltration from benign data transfers, potentially resulting in an unacceptable false positive rate. Threat partitioning induces an interval of time in which to detect anomalies, while potentially simplifying the benign traffic in which the anomalies should be searched *and* potentially simplifying the anomalies themselves.

3.2 Network Packet Data Structuring

Data structuring involves the discovery of relationships and abstractions within the raw packet data (PCAP), and encoding those relationships in a manner conducive for propositional machine learning. Using the netflow structure discussed in Section 2.2 as a base, we would like to encode prior knowledge inherent to the known threat traffic that we collect.

We observe that distributed threat traffic, for example bot generated DDoS traffic, comprise flows which will overlap in time as they arrive at their target. [2] To explicitly represent this *inter-flow* temporal relationship, named flow concurrency [23], we first create a new netflow feature to represent the relationship. Then for any flow, we count the number of similar flows arriving alongside it and set this count as the value of the new netflow feature.

More formally, two flows i and j sharing a common destination IP address, destination port, and protocol overlap in time if

$$\min[t_s(i) + d(i), t_s(j) + d(j)] > \max[t_s(i), t_s(j)], \tag{1}$$

where t_s represents a flow's start time and d represents the duration of a flow at the time of overlap calculation.

As bot generated traffic tends to exhibit similar, deterministic behavioral patterns, if two flows overlap in time but exhibit wildly different behavioral characteristics, those flows may be uncorrelated, simply by chance having arrived at the same place at a similar time. To capture inter-flow behavioral similarity, we devise a metric based on netflow aggregate features. We define this similarity metric as

$$Sim = 1 - \left(\sum_{a \in \mathbf{a}} \frac{|a(i) - a(j)|}{max[a(i), a(j), 1]} \right) / |\mathbf{a}|, \tag{2}$$

where a is an aggregate feature in the set of aggregates $\mathbf{a}$, and $|\mathbf{a}|$ is the number of aggregates in the set. We assume that aggregates are non-negative. Equation 2 is a normalized L1 norm, and two flows are considered similar if $Sim > \epsilon$, where ϵ is a threshold value.

For each flow, an aggregate feature is added representing the count of flows concurrent to it. Flow concurrency may be a useful relation for many different threat and benign behavioral profiles. In addition to DDoS traffic similarities, distributed attacks with deterministic c&c may exhibit this determinism across all nodes which participate in the attack [7].

While flow concurrency captures an inter-flow relationship, there may also exist relationships within a single flow. Such *intra-flow* relationships can be used to explicitly capture non-standard behavior which may be observed in threat variants. For example, as discussed in Section 2.4, SYN packets should not contain a payload, but yet SYN flood variants may purposely include one to avoid detection.

A SYN packet's payload will manifest in the packet byte feature of a netflow, which as an aggregate, the relationship to which packet the payload belongs is lost. To explicitly represent the presence of a SYN packet payload, we add a new feature to represent it. If the flow's SYN packet contains a payload, we first subtract the payload's size from the packet byte aggregate, then we set as the new feature's value the payload size. For SYN packets without a payload, this new feature's value will be 0.

Another intra-flow relationship we observe is that a TCP stream should always begin with a SYN packet. To represent this relationship, we create two additional features, representing a copy of the netflow's packet bytes and packet count, respectively. Then for any TCP netflow which does not start with a SYN packet, the packet byte and count values of the flow are copied to the two new features. [3] Unlike the payload relationship given above, here we do not modify the original features, as we do not know how large the missing SYN packet should be. Non-zero values for these two new features serve to 'flag' a TCP flow as being non-standard.

In general, we can represent an intra-flow relationship by finding the aggregate feature or features which implicitly captures it, creating a new feature or features to explicitly represent the relationship, and then computing new values for the newly created feature or features. While this technique has the potential to greatly increase the number of features, our netflow structure with the additional features mentioned here gives us 12 in total, which is still quite low. Figure 2 illustrates the features in our netflow structure.

3.3 Detection Models

It has been long known in AI and ML literature that ensemble models routinely perform better than a single model on large tasks such as recommendation or information retrieval [6, 19]. For example,

[2] This characteristic also appears in single-bot SYN floods when source address spoofing is utilized.

[3] For TCP streams comprising multiple flows, if the SYN packet is missing from the first flow, all subsequent flows will also have their packet byte and count values copied.

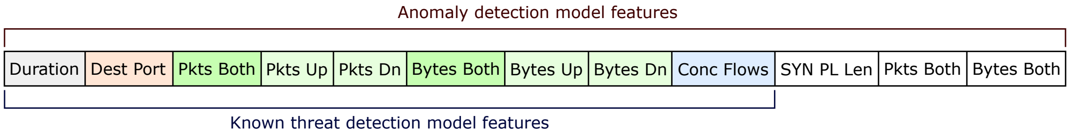

Figure 2: Illustration of our netflow structure. 'Dest Port' is the netflow's destination port number. The next six features refer to the number of packets and total packet bytes which a flow comprises, representing these counts in the upstream and downstream ('Up' and 'Dn'), and bi-directional ('Both'). 'Conc Flows' is a count representing flow concurrency. The intra-flow features, 'SYN PL Len', 'Pkts Both', and 'Bytes Both' represent SYN packet payload, and the number of bi-directional packets and bytes in a TCP stream which does not begin with a SYN packet, respectively.

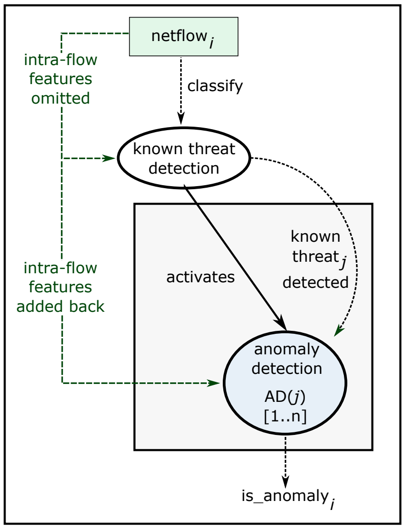

Figure 3: High-level illustration of our proposed intrusion detection system, as a plate model. The known threat detection model can detect n known threat classes, where each class maps to an anomaly detection model.

learning individual models for specific network traffic classes would allow for learning expressive representations. Hence, we propose an ensemble model to handle detection of known and anomalous threat behavior, comprising one model for known threat detection of n threat classes, and n models for anomaly detection.

Figure 3 illustrates our proposed IDS as a plate model, with a 1-to-many relationship between the known threat detection model and $[1..n]$ anomaly detection models. [4] Here, we have one anomaly model for each threat class represented in the known threat model. The steps are as follows: a netflow i is passed to the known threat model with its intra-flow features omitted. If the classification of netflow i is threat class j, the respective anomaly model $AD(j)$ is activated, and will classify netflow i with its intra-flow features restored. If netflow i is not classified as an anomaly by the anomaly model, then the known threat model's classification is used.

We omit the intra-flow features from the known threat model as these features represent behavior not common to a known threat class. Detecting such behavior is the job of the anomaly model, hence these intra-flow features are added back to any netflow which is passed to an anomaly model for classification.

For our proposed IDS, all detection models in the ensemble are Gaussian Mixture Models (GMMs). A GMM is a soft-clustering model which gives a probability that a netflow belongs to each of the learned clusters. This is an unsupervised task, with the only labeling required in the form of a small dataset of representative netflows from each known threat class, so we can map each one to the cluster that represents it in the known threat model.

A related model, variational Bayesian GMMs [3], learns the number of clusters using a Dirichlet process prior and upper bound on cluster count. This introduces a hyper-parameter to control mixture weights, which ultimately determines which clusters will be included in the final model. Using a traditional GMM for the known threat model, we learn one cluster for each threat class, avoiding the need to blindly tune a hyper-parameter.

To support anomaly detection in a GMM model, we compute the likelihood that a netflow was 'generated' from the model. This is roughly analogous to bounding the clusters using a distance metric to define an anomalous space. For spherical clusters (and their hard-clustering analogue) the bound resembles a radius [24]. In our setting we use full covariance, providing variance per feature and correlations between them, individually learned per cluster.

All GMM models in our ensemble support anomaly detection. For each model, we compute a threshold τ_b to determine if a netflow should *not* be classified to any of the represented clusters by taking the mean of log-likelihood values of netflow classifications from an unlabeled validation dataset, and using the three-sigma rule, or three deviations from the mean, to soften the threshold. As log-likelihood values strictly decrease as netflows become less likely to belong to any clusters in the model, we only need to represent a single threshold value. More formally, a netflow i is classified as not belonging to any cluster in the model if

$$ll_i < \tau_b, \tag{3}$$

where

$$\tau_b = \mu(d_t) - 3\sigma(d_t). \tag{4}$$

In Equation 3, ll_i is the log-likelihood of netflow i, and in Equation 4, $\mu(d_t)$ and $\sigma(d_t)$ are the mean and standard deviation of log-likelihood values from validation dataset d_t, respectively.

The known threat detection model does *not* have a cluster representing benign traffic, instead comprising n clusters where n is the number of unique known threat classes (see Figure 3). Here, anomaly detection is inverted, with benign traffic classified as the 'anomalies'. As discussed in Section 3.1, benign traffic collection

[4] We use the terms 'known threat detection model' and 'known threat model' interchangeably. Same with 'anomaly detection model' and 'anomaly model'.

is often a difficult problem, and this approach alleviates having to collect any benign traffic under nominal network conditions. Thus, the known threat model is trained only on known threat traffic.

An anomaly detection model comprises clusters representing known threat behavior also captured by the known threat detection model, and benign traffic behavior in the presence of the detected threat class. Therefore, we treat the model as a one-class classifier, with a cluster count reasonable to capture known threat and benign traffic behavior. While a cluster count of two is straightforward, we can increase this to further capture more fine grained behavioral nuances for both known threat and benign traffic. Each anomaly model is trained using known threat traffic from a single threat class along with benign traffic collected concurrently with it.

A validation dataset used for computing τ_b (Equation 4) for the known threat model should be overly sensitive to benign traffic to reduce known threat false positives. Netflows which have a high likelihood of belonging to the model should give us this desired characteristic. One such dataset is the training data itself, thus for the known threat detection model, we use the training dataset as the validation dataset for computing τ_b. Validation datasets for the anomaly detection models should comprise relevant known threat and benign behavior incorporating concept drift. Such behavioral variations help to push τ_b to a value in which these variations are not considered anomalous, potentially resulting in a longer service time before the model needs to be retrained.

Because classification is a discrete-time task, we need some way to keep an anomaly detection model active when faced with a mix of incoming known threat and benign traffic. We take a rolling-window approach in which multiple netflows must classify as a known threat of the same class within a set duration to keep the respective anomaly model active. This type of rolling-window may also help to prevent spurious known threat detection by smoothing incorrect classifications.

4 EXPERIMENTS

In discussing our experiments, we first describe the datasets used and experimental methodology. We then present the results, providing analysis on detection performance for both DaaSS and SYN flood threat classes.

4.1 Datasets

We curated four datasets for our experimentation, with each one testing our proposed IDS in different ways. Finding relevant PCAP data which captures DaaSS or SYN flood variants is inherently difficult, not to mention that externally curated datasets may contain zero-days or other unknown anomalies which we may not be aware of. Thus, all four datasets we curated in-house using a network data generation framework we developed named eMews [22], enabling the creation of representative network traffic free from unknown anomalies. Information about each dataset is given in Table 1.

Network traffic for all four datasets was captured on similar network topology, with the targeted network consisting of an HTTPS server, SSH server, and two workstations. Benign and threat traffic originate from hosts which reside in various local-area networks outside the targeted network. The DDoS attack used as the known threat behavior for the first two datasets is a reflection attack on the HTTPS server of the targeted network. A reflection attack occurs when an attacker, in this case a botnet, sends low volume traffic to a target to induce a high volume outbound flood from the target. This type of behavior was picked as it can resemble legitimate benign HTTPS traffic, potentially resulting in a more difficult detection task.

Scenario	Cap Dur	KT Dur	Cn Drft	# Bn Flows	# KT Flows	# An Flows
NDP-Train	2 h	2 h	-	4800	439358	-
NDP-DDoS-1	30 m	30 m	N	1101	111051	-
NDP-DDoS-2	30 m	30 m	Y	1261	116206	-
NDP-DaaSS-1	30 m	30 m	N	1319	120419	7
NDP-DaaSS-2	30 m	30 m	Y	1265	123854	21
TS2-Train	1 h	20 m	-	1505	2240	-
TS2-DDoS-1	30 m	10 m	N	2343	1123	-
TS2-DDoS-2	30 m	10 m	Y	2465	1120	-
TS2-DaaSS-1	30 m	10 m	N	2222	1120	5
SYN-Train	30 m	1.3 m	-	94	135251	-
SYN-Base-1	10 m	1.3 m	N	752	140652	-
SYN-Var-1	12 m	1.5 m	Y	731	135000	254
SYN-Var-2	12 m	1.5 m	Y	749	252	280800
MT-Base-1	2 h	5 m 0.8 m	Y	10101	608 81251	-

Table 1: Netflow scenarios used for experimentation. 'Cap Dur' and 'KT Dur' refer to a scenario's capture duration and known threat duration, respectively. Duration is measured in hours (h) or minutes (m). 'Cn Drft' refers to concept drift, with scenarios labeled 'Y' incorporating varying network traffic behavior compared to the training scenario. '# Bn Flows', '# KT Flows', and '# An Flows' refer to the number of benign netflows, the number of known threat netflows, and the number of anomaly netflows in a scenario, respectively.

The first dataset, named *NDP*, was curated in 2018 and contains scenarios comprising both DDoS and DaaSS attacks [24]. We define a *scenario* as a set of netflows parsed from a single PCAP capture, which comprise our training and testing sets. The underlying threat of the DaaSS attacks captured is an SSH-based insider data exfiltration, originating from one or more of the workstations within the targeted network. NDP scenarios are unique in that the threat is present throughout the entirety of each scenario.

The second dataset, named *TS2*, was curated in 2021 and also captures DDoS and DaaSS attacks [23]. The HTTPS server in the targeted network for TS2 scenarios is configured to utilize server-side connection persistence, an often used option for web servers. Unlike with the NDP dataset, the underlying threat of the DaaSS attack captured in the TS2 dataset originates externally from the targeted network. While the training scenario duration is 1 hour, for training we only use the 20 minute duration in which the known threat behavior is present, as the rest of the scenario is benign traffic which we do not need for training. Table 1 lists flow counts within the known threat duration for all training scenarios.

The third dataset, which we aptly named *SYN*, captures SYN floods and variants. The first scenario is a SYN flood which resembles a slightly more aggressive form of the SYN flood used for training. The other two are variants: a large payload SYN flood (*SYN-Var-1*), and a large payload SYN flood with unsolicited TCP packets (*SYN-Var-2*), similar to the SYN flood attack described by Imperva [2]. The SYN training scenario duration is 30 minutes, but for training we only use the 1.3 minute duration in which the known threat behavior is present, as the rest of the scenario is benign traffic which we do not need for training.

The fourth dataset comprises one scenario, *MT-Base-1*, which captures two known threat behaviors: a DDoS of the type found in the TS2 scenarios, and a SYN flood of the type found in *SYN-Base-1*. The DDoS starts about 15 minutes into the scenario, and is present for 5 minutes. Then, an hour into the scenario, the SYN flood starts, and is present for 0.8 minutes.

4.2 Methodology

When parsing PCAP data to netflow scenarios, inactive and active flow timeouts were set to 15 seconds and 60 seconds, respectively, to match default values commonly used in industrial netflow systems. Similarity epsilon ϵ for flow concurrency was set to 0.99, based on an expectation that DDoS flows will not vary much from one another. Aggregates used to compute flow similarity were the unidirectional packet count and bytes in the upstream and downstream.

Netflow scenarios *NDP-DDoS-1*, *NDP-DDoS-2*, *TS2-DDoS-1*, *TS2-DDoS-2*, *SYN-Base-1*, *SYN-Var-1*, and *MT-Base-1* were used to measure IDS performance in the absence of any underlying unknown threat behavior. These scenarios can give insights into how well our proposed IDS can discriminate between benign and known threat traffic. Netflow scenarios *NDP-DaaSS-1*, *NDP-DaaSS-2*, *TS2-DaaSS-1*, and *SYN-Var-2* were used to give insights into detection performance of unknown threat behavior.

Netflow scenario *MT-Base-1* was used to test our proposed IDS on multiple known threat classes in a single scenario. To train our proposed IDS, we used both the TS2 and SYN training scenarios by combining them into a single training scenario.

We set the rolling window of our proposed IDS to 17 seconds, which will keep anomaly detection active 17 seconds beyond detection of known threat traffic (classification of known threat netflows). We selected 17 seconds due to our netflow inactive timeout being 15 seconds, plus an additional 2 seconds to compensate for small gaps in arrival time between consecutive netflows which time out due to this inactive timeout.

Initial cluster weights and means for our GMM-based detection models were computed using k-Means++ [1]. Each learned cluster in the GMM utilized its own covariance matrix, representing per-feature variance and inter-feature correlation. For the known threat detection model, we set the number of clusters to learn to match the count of known threat classes represented in our scenarios for each dataset. For the anomaly detection models, the number of clusters to learn was set to 15, based on an upper bound of expected behavioral variations within the known threat and benign traffic.

Comparing our proposed IDS against another IDS required the other IDS to support both multi-class threat detection with anomaly detection, and netflow data to represent the inter-flow and intra-flow relationships. The only IDS we are aware of which meets these constraints comes from our prior work in DaaSS detection [24]. This IDS, which we will call *N1*, comprises a single hard-clustering model capable of multi-class threat detection with anomaly detection, and supports netflow data.

N1 was initialized with 2 cluster centroids to represent the known threat and benign behavior, using the initialization method introduced by Cao et. al. [4]. A scaling hyper-parameter, α_r, is used to adjust sensitivity to anomalies by scaling the radius learned to bound each cluster. We set $\alpha_r = 1.5$, the value which gave the best results from our prior work [24].

We tested both IDS through all the test scenarios in both datasets multiple times, taking as the result the best run. The performance metrics we focused on were those which illustrated a model's ability to limit false positives.

4.3 Experimental Results and Discussion

For all scenarios, our rolling window approach kept anomaly detection active in our proposed IDS during the periods of known threat activity, without any deactivation of anomaly detection during those periods. Furthermore, our proposed IDS did not produce any known threat false positives in the absence of known threat traffic, implying that it can operate for long periods of time without false alarms.

Using validation scenarios to automatically set model sensitivity seemed to work well for keeping false positives down, both in terms of known threat and anomaly false positives. We further elaborate on our known threat detection results in the next section, then discuss anomaly detection results.

4.3.1 Known Threat Behavior Detection.

Referring to our threat detection results in Table 2, our proposed IDS produced no known threat false positives for all TS2, SYN, and MT scenarios, which is important as any false alarms here would unnecessarily activate underlying anomaly detection.

For the TS2 scenarios, our proposed IDS performed similarly to N1, with N1's known threat recall slightly higher. In the precision vs recall tradeoff, we would prefer to miss some known threat traffic rather than produce false alarms. Specifically in regard to *TS2-DaaSS-1*, N1's benign accuracy dropped below 100%, misclassifying a benign netflow as an anomaly. This misclassified benign netflow arrived about 28 seconds after the last known threat netflow, but N1 cannot constrain when it searches for anomalies, hence the false alarm.

For the SYN scenarios, our proposed IDS showed strong performance, with zero known threat false positives for all scenarios. Here, N1 struggled to discriminate between benign and known threat traffic, manifesting in hundreds of false alarms per scenario and benign accuracy in the 30% range. Experimenting with different values for N1's α_r scaling hyper-parameter did not help improve the benign accuracy of the trained model. We also experimented with using N1 as an OCC model for known threat detection, treating benign traffic as anomalous in much the same way as our GMM known threat detection model does. N1 did not show any improvement, suggesting that using a GMM for known threat detection leads to better discrimination between benign and known threat traffic.

Focusing on scenario *MT-Base-1*, our proposed IDS had zero known threat false positives throughout the scenario, which considering there were two attacks present, shows that our proposed

	Proposed IDS							N1 (α_r = 1.5)						
	Known Threat				Underlying Anomaly			Known Threat				Underlying Anomaly		
Scenario	# FP	Bn Acc	Prc	Recall	# FP	Prc	Recall	# FP	Bn Acc	Prc	Recall	# FP	Prc	Recall
NDP-DDoS-1	**170**	**84.56%**	99.85%	99.32%	0			338	69.3%	99.7%	99.82%	0		
NDP-DDoS-2	**167**	**86.76%**	99.86%	99.4%	6			370	70.58%	99.68%	99.79%	**1**		
NDP-DaaSS-1	**205**	**84.54%**	99.83%	98.64%	9	43.75%	100%	404	69.29%	99.66%	99.73%	**1**	**87.5%**	100%
NDP-DaaSS-2	**178**	**86.16%**	99.85%	98.48%	10	58.33%	66.67%	390	69.09%	99.68%	99.65%	**1**	**94.44%**	**80.95%**
TS2-DDoS-1	0	100%	100%	97.06%	0			0	100%	100%	99.73%	0		
TS2-DDoS-2	0	100%	100%	97.05%	0			0	100%	100%	**100%**	0		
TS2-DaaSS-1	0	**100%**	100%	96.79%	1	75%	60%	0	99.95%	100%	**100%**	1	75%	60%
SYN-Base-1	**0**	**100%**	100%	99.46%	**1**			521	29.26%	99.63%	100%	11		
SYN-Var-1	**0**	**100%**	100%	99.44%	**0**	**100%**	**100%**	501	30.78%	99.63%	100%	5	0%	0%
SYN-Var-2	**0**	**100%**	100%	99.45%	**2**	**99.99%**	**100%**	513	31.11%	99.64%	100%	3	0%	0%
MT-Base-1	**0**	**100%**	**100%**	99.21%	0/1			7185	28.87%	91.93%	100%	**0**		

Table 2: Performance of our proposed IDS compared to the N1 hard-clustering model. For scenarios not containing anomalies, underlying anomaly precision and recall are not applicable. '# FP' represents false positive counts, 'Bn Acc' represents benign accuracy, and 'Prc' represents precision.

IDS can keep the known threat false positives at zero in the face of multiple threats. N1 performed the worst here of any scenario. While N1 classified all the DDoS threat traffic correctly, 71.13% of benign traffic was classified as a SYN flood, and suggests that N1 will consistently give false alarms of SYN flood attacks if trained to detect them.

For the NDP scenarios, while our proposed IDS had a significantly lower number of false positives and a significantly higher benign accuracy than N1, our IDS still produced hundreds of false alarms. This again suggests that a GMM may be better suited for known threat detection when compared to hard clustering.

Interestingly, our proposed IDS showed slightly better benign accuracy on NDP scenarios with concept drift compared to NDP scenarios without. The known threat detection model may be naturally suited to handling concept drift in benign traffic, as the model does not know what benign traffic looks like in general. This ties into model sensitivity, which our approach to automatically set at training time seems to work well.

4.3.2 Underlying Anomaly Detection Results. For the TS2 scenarios, both our proposed IDS and N1 performed well with only 1 anomaly false positive occurring for each IDS in *TS2-DaaSS-1*. However, N1's false positive occurred about 2 minutes after the DDoS had ceased, which would result in a false alarm. Because our proposed IDS only searches for anomalies during the presence of known threat behavior, for this scenario no change in classification would occur as anomalies are already present.

For the SYN scenarios, our proposed IDS performed very well, with perfect anomaly classification on *SYN-Var-1*. The N1 model struggled on all scenarios, not only with a higher count of anomaly false positives, but N1 could not actually detect any anomalies. For scenarios *SYN-Var-1*, and *SYN-Var-2* in which the variants are represented as intra-flow features, these features do not seem to have an effect on N1's classification, rendering it blind to SYN flood variants. Furthermore, for all SYN scenarios, all anomaly false positives from N1 occurred when the SYN flood was not active, resulting in false alarms.

Focusing on scenario *MT-Base-1*, our proposed IDS had zero anomaly false positives when the DDoS was present, and one false positive when the SYN flood was present. This seems to align with our results for the TS2 and SYN scenarios which did not contain anomalies, even though *MT-Base-1* was curated separately. N1 has one anomaly representation for all known threat classes it represents, which here was not an issue as N1 produced no anomaly false positives.

Our proposed IDS did not perform as well on the NDP scenarios, with between 6 - 10 anomaly false positives in all but *NDP-DDoS-1*, which had zero. Our IDS was able to detect the unknown threat behavior in *NDP-DaaSS-1* and *NDP-DaaSS-2*, with *NDP-DaaSS-1* having perfect recall. Overall N1 performed well with the NDP scenarios, out-performing our IDS in three out of four scenarios.

Concept drift did not seem to have much effect on our proposed IDS in terms of anomaly false positives. Also interesting to note is the duration in which the anomaly detection models were active did not seem to affect the anomaly false positive counts. In other words, longer duration of anomaly detection does not appear to correlate with more false positives. This may suggest that anomaly false positives tend to originate at the fringes of anomaly detection, either at the beginning of activation or at the end.

5 LIMITATIONS AND FUTURE WORK

A key point brought up by Sommer and Paxson [27] was the lack of relevant datasets for anomaly model training and testing, and difficulty in curating or creating them. This includes issues with benchmark datasets, such as their age and curation methods. While there are efforts to develop realistic network data generation methods specifically to address these issues [22, 26], generating representative benign traffic is in general a difficult task, hence our approach of minimizing the amount we need to collect.

The NDP dataset was difficult due to benign and known threat traffic exhibiting very similar behavioral characteristics. An interesting direction for future work involves additional inter-flow feature engineering to explicitly represent subtle differences in behavior between the known threat class and benign traffic. Ideally, any additional inter-flow features should be useful for a wide range of threat classes.

As the number of threat classes represented in the known threat model increases, the number of anomaly detection models also increases. If considering multiple concurrent attacks from threats belonging to different classes, we would need additional anomaly detection models to represent the different combinations, possibly resulting in combinatorial explosion regarding the datasets we would need to curate. Using network traffic generation techniques such as eMews may help to make this task more scalable [22].

Our proposed IDS is designed to detect zero-day threats as long as the zero-day encompasses some known threat behavior. If malicious traffic does not first match any known threat class, it will go undetected by design. While this is a limitation when compared to purely anomaly-based IDS, we point out that threat partitioning provides context to the anomalies detected due to the known behavior observed. With a purely anomaly-based IDS, detected anomalies may need inspection to determine if they are actually threats. With our proposed IDS, anomaly false negatives result in the known threat behavior being incorrectly classified as a base threat class, when in reality it is a zero-day variant.

6 CONCLUSION

In this work, we developed and demonstrated through experimentation an anomaly-based network intrusion detection system which utilizes a method we call threat partitioning to scope the search for threats, both in terms of the threats being searched for, and the time intervals in which the IDS searches for them. Our detection results on two threat classes show promise that our IDS may be practical for industrial adoption and deployment.

ACKNOWLEDGMENTS

SN acknowledges the support by ARO award W911NF2010224. This material is also based upon the work partially supported by the National Science Foundation under Award No. (FAIN): DGE-1723602. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARO or National Science Foundation.

REFERENCES

[1] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, Louisiana) *(SODA '07)*. Society for Industrial and Applied Mathematics, USA, 1027–1035.

[2] Johnathan Azaria. 2020. DDoS Attacks Grow More Sophisticated as Imperva Mitigates Largest Attack. https://www.imperva.com/blog/ddos-attacks-grow-more-sophisticated-as-imperva-mitigates-largest-attack/

[3] David M. Blei and Michael I. Jordan. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis* 1, 1 (2006), 121 – 143.

[4] Fuyuan Cao, Jiye Liang, and Liang Bai. 2009. A New Initialization Method for Categorical Data Clustering. *Expert Syst. Appl.* 36, 7 (Sept. 2009), 10223–10228.

[5] D. E. Denning. 1987. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering* SE-13, 2 (Feb 1987), 222–232.

[6] Devendra Singh Dhami, Siwen Yan, Gautam Kunapuli, and Sriraam Natarajan. 2022. Non-parametric Learning of Embeddings for Relational Data Using Gaifman Locality Theorem. In *Inductive Logic Programming*, Nikos Katzouris and Alexander Artikis (Eds.). Springer International Publishing, Cham, 95–110.

[7] Christian J. Dietrich, Christian Rossow, and Norbert Pohlmann. 2013. CoCoSpot: Clustering and Recognizing Botnet Command and Control Channels Using Traffic Analysis. *Comput. Netw.* 57, 2 (Feb. 2013), 475–486.

[8] Yebo Feng, Jun Li, Lei Jiao, and Xintao Wu. 2019. BotFlowMon: Learning-based, Content-Agnostic Identification of Social Bot Traffic Flows. In *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, Washington, D.C., 169–177.

[9] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An Empirical Comparison of Botnet Detection Methods. *Computers & Security* 45 (2014), 100–123.

[10] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. 2014. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials* 16, 4 (2014), 2037–2064.

[11] Zhexue Huang. 1997. Clustering Large Data Sets with Mixed Numeric and Categorical Values. In *The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*. World Scientific, Singapore, 21–34.

[12] Scott Ikeda. 2022. 3.7 Million FlexBooker Accounts Leaked to Hacker Forum After DDoS Attack. *CPO Magazine* (2022). https://www.cpomagazine.com/cyber-security/3-7-million-flexbooker-accounts-leaked-to-hacker-forum-after-ddos-attack/

[13] Mohammad Karami and Damon McCoy. 2013. Understanding the emerging threat of DDoS-As-a-service. In *Proceedings of the 6th USENIX Conference on Large-Scale Exploits and Emergent Threats* (Washington, D.C.) *(LEET'13)*. USENIX Association, USA, 8.

[14] Kaspersky Lab. 2016. *Corporate IT Security Risks Survey.* Technical Report. Kaspersky Lab.

[15] Tushar Khot, Sriraam Natarajan, and Jude Shavlik. 2014. Relational one-class classification: a non-parametric approach. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. AAAI Press, Québec City, Québec, Canada, 2453–2459.

[16] Jan Larsen, Lars Kai Hansen, Anna Szymkowiak Have, Torben Christiansen, and Thomas Kolenda. 2002. Webmining: Learning from the World Wide Web. *Comput. Stat. Data Anal.* 38, 4 (feb 2002), 517–532.

[17] Rodney A. Martin. 2007. Unsupervised Anomaly Detection and Diagnosis for Liquid Rocket Engine Propulsion. In *2007 IEEE Aerospace Conference*. IEEE, Big Sky, MT, USA, 1–15.

[18] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M. Thuraisingham. 2011. Classification and Novel Class Detection in Concept-Drifting Data Streams Under Time Constraints. *IEEE Trans. on Knowl. and Data Eng.* 23, 6 (June 2011), 859–874.

[19] Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude Shavlik. 2012. Gradient-Based Boosting for Statistical Relational Learning: The Relational Dependency Network Case. *Machine Learning* 86, 1 (01 Jan 2012), 25–56.

[20] Vern Paxson. 1999. Bro: A System for Detecting Network Intruders in Real-Time. *Comput. Netw.* 31, 23–24 (Dec 1999), 2435–2463.

[21] J. Postel. 1981. Transmission Control Protocol. RFC 793. https://www.rfc-editor.org/info/rfc793

[22] Brian Ricks, Patrick Tague, and Bhavani Thuraisingham. 2018. Large-Scale Realistic Network Data Generation on a Budget. In *19th International Conference on Information Reuse and Integration (IRI)*. IEEE, Salt Lake City, Utah, USA, 23–30.

[23] Brian Ricks, Patrick Tague, and Bhavani Thuraisingham. 2021. DDoS-as-a-Smokescreen: Leveraging Netflow Concurrency and Segmentation for Faster Detection. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, Los Alamitos, CA, USA, 217–224.

[24] Brian Ricks, Bhavani Thuraisingham, and Patrick Tague. 2018. Lifting the Smokescreen: Detecting Underlying Anomalies During a DDoS Attack. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, Miami, FL, USA, 130–135.

[25] Martin Roesch. 1999. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration* (Seattle, Washington) *(LISA '99)*. USENIX Association, USA, 229–238.

[26] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security* 31, 3 (2012), 357 – 374.

[27] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP '10)*. IEEE Computer Society, Washington, DC, USA, 305–316.

[28] StormWall. 2023. H1 2023 in Review: DDoS Attacks Report by StormWall. https://stormwall.network/ddos-report-stormwall-h1-2023

[29] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. 2009. Intrusion Detection by Machine Learning: A Review. *Expert Systems with Applications* 36, 10 (2009), 11994–12000.

[30] Paul Wagenseil. 2011. Sony Blames Anonymous for PlayStation Network Attack. http://www.nbcnews.com/id/42909386/ns/technology_and_science-security/t/sony-blames-anonymous-playstation-network-attack/.