APEX: Characterizing Attack Behaviors from Network Anomalies

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

26-01-2022 / 01-02-2022

CITATION

Kalupahana Liyanage, Kushan Sudheera; Tian, Zixu; Divakaran, Dinil Mon; Chan, Mun Choon; Gurusamy, Mohan (2022): APEX: Characterizing Attack Behaviors from Network Anomalies. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.19070195.v1>

DOI

[10.36227/techrxiv.19070195.v1](https://doi.org/10.36227/techrxiv.19070195.v1)

APEX: Characterizing Attack Behaviors from Network Anomalies

Kushan Sudheera Kalupahana Liyanage^{*} , Zixu Tian^{*} , Dinil Mon Divakaran^{ },
Mun Choon Chan^{ }, and Mohan Gurusamy^{ }

Abstract—Networks regularly face various threats and attacks that manifest in their communication traffic. Recent works proposed unsupervised approaches, e.g., using a variational autoencoder, that are not only effective in detecting anomalies in network traffic, but also practical as they do not require ground truth or labeled data. However, the problem of characterizing anomalies into different attack behaviors is still less explored; in this work, we study this specific problem. We develop **APEX**, a framework that employs data mining approaches in a semi-supervised way to extract the attack patterns from anomalous traffic and links them to specific attack types. **APEX** comprises two levels of mining; the first level extracts patterns in anomalous network flows, and the second level characterizes behaviors in the extracted patterns into four different attack classes. We carry out extensive experiments on real network traces obtained from the MAWI traffic archive. The evaluations demonstrate that **APEX** is effective in extracting distinguishable behaviors of network attacks from anomalous traffic, which we believe, provides useful insights to security analysts investigating the anomalies.

Index Terms—Data Mining, Network Security, Anomalies.

I. INTRODUCTION

NETWORKS are facing an increasing number of threats, attacks and data breaches in recent times [1]. These are often carried out in different stages such as reconnaissance (scanning), brute-force login attempts, malware download, C&C (command and control) communications, data exfiltration, DDoS attacks, etc. [2], [3]; and they manifest in network traffic. Hence, analyzing network traffic helps in detecting and identifying the threats and attacks organizations face.

Past works have attempted to detect (often specific) attacks using supervised learning approaches that model the attack detection as a binary classification problem [4]–[6], where labeled data of two classes of network traffic—normal and anomalous—are gathered and provided for training the models. However, network traffic (of, say, enterprises) tends to be noisy due to the large number of users, evolving landscape of applications and protocols (e.g., adoption of TLS 1.3 and

HTTP/3), increasing use of new smart devices, etc. Therefore, the definition of ‘normal’ changes with time and space (i.e., the network where the model is deployed). Besides, labeling each traffic flow of a network as normal/anomalous is labour-intensive and unsustainable for model maintenance (retraining). Therefore, researchers have proposed unsupervised approaches for network anomaly detection, with recent works exploring unsupervised deep learning models [7], [8].

While unsupervised models are effective in detecting anomalies, they do not explain what these anomalies are. The anomalies could be (due to) scans, low-rate login attempts, floods, etc. Without such explanations, an analyst who investigates the anomalies would find it hard to understand the nature of the threat that is affecting the network. For easing the investigation and connecting the dots between the detected anomalies, a security solution should provide more information and context regarding the detected anomalies. In this work, we focus on this specific problem of characterizing attack behaviors from network anomalies.

We propose and develop **APEX** (**A**ttack **P**attern **E**xtractor), a novel semi-supervised framework for analyzing, detecting and identifying attack behaviors from network anomalies. Specifically, we consider an existing anomaly detection solution that detects and ranks anomalies using an unsupervised model. **APEX** takes the top-ranked anomalies from such a solution and proceeds to function in multiple phases. First, it defines and extracts features from anomalies that capture different kinds of network attack behaviors. Subsequently, **APEX** employs an efficient and effective data mining technique called FIM (frequent itemset mining) at two levels to extract and group anomalies into a small number of attack-behavioral patterns. The first level of mining extracts attack patterns from five-tuple anomalous flows in an unsupervised way. The mined patterns are then grouped according to attack types using labeled information. Subsequently, another level of mining (second level) is carried out on the grouped patterns to extract behavioral patterns of different attack classes. These behavioral patterns can be used to identify different attack types from anomalies and also provide useful insights to a security analyst. We carry out comprehensive experiments using packet-level real network traffic obtained from the MAWI archives [9]. **APEX** achieves comparable performance with the best-performing supervised model, while providing behavioral patterns that characterize and help visualize the different attacks.

We present an overview of **APEX** framework in Section III. In Section IV, we present the **APEX** framework in detail

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible

K. S. K. Liyanage was with Dept. of Computer Science, National University of Singapore, Singapore and now with Dept. of Electrical and Information Engineering, University of Ruhuna, Sri Lanka (e-mail: kushan@eie.ru.ac.lk).

Z. Tian and M. Gurusamy are with Dept. of Electrical and Computer Engineering, National University of Singapore, Singapore (e-mail: tianzixu@u.nus.edu, elegm@nus.edu.sg)

D. M. Divakaran is with Trustwave, Singapore (e-mail: dinil.divakaran@trustwave.com).

M. C. Chan is with Dept. of Computer Science, National University of Singapore, Singapore (e-mail: chanmc@comp.nus.edu.sg)

^{*} Zixu Tian shares equal contribution with the first author for this work.

depicting the features and the extraction of behavioral patterns. Section V presents the experiments and analyzes the results.

II. BACKGROUND AND RELATED WORKS

Network anomaly detection has been an important problem for decades; and many past works have explored different statistical approaches for this purpose [10]–[12]. One well-known approach is to apply supervised learning models. For example, Bilge et al. [4] proposed a botnet detection system by training a Random Forest (RF) classifier on features extracted from NetFlow data. Although supervised learning is an effective approach, it is based on the ground truth of existing intrusion. Whereas the ever-evolving nature of networks (home, enterprises, etc.) as well as the threat landscape lead to newer threats and attacks. This trend may shift the decision boundary differentiating the pre-defined attack and benign traffic, and thus reduce the effectiveness of supervised anomaly detectors. More importantly, supervised algorithms face deployment roadblock, since they are dependent on the availability of ground truth on both classes of network flows, normal and anomalous. Precise labeling of raw network traffic flows is labour-intensive and impractical [13], and this challenges the utility of supervised approaches in real deployments. Motivated by this challenge, recently Jan et al. attempted to overcome it using GAN (Generative Adversarial Network) to augment limited labeled data, although for the specific detection of botnet attacks on web servers [14].

In comparison, unsupervised approaches are often more practical than the supervised approaches for anomaly detection, given the continuous evolution of network traffic and unavailability of ground truth. One common unsupervised approach is clustering [15], which is used to group data with high similarities, and then separate the anomalies in terms of the marginal groups. For example, classical measures such as entropy-based data analysis was proposed in [16] to cluster anomalies in network data. Yen et al. [17] presented an approach of detecting the anomalous behaviors by clustering the activity logs of enterprise networks.

Unsupervised deep learning models such as Autoencoder [18], Variational Autoencoder (VAE) [19] achieve robust and good performance in dealing with complex and noisy data, leading to the recent exploration of these models for network anomaly detection. For example, the authors in [20] leveraged VAE in learning latent representation of high-dimensional and sparse network data, to improve anomaly detection performance over a series of one-class classification algorithms. Yet another recent work, GEE [7], presents an unsupervised anomaly detection framework using VAE, with an additional gradient-based fingerprinting technique that attempted in explaining the anomalies in network traffic.

In spite of the promising performance, most existing anomaly detection solutions tend to stop at binary classification, without looking deeper into the exact attributions of the anomalies detected, e.g., the specific attack types or different behavior patterns of mixed anomalies. Even though there are works like [21] and [22] that explore ways to explain and comprehend anomalies, those focusing on exact categorization of anomalies to attack classes are rare.

Data mining is a class of approaches for identifying data patterns, and Frequent itemset mining (FIM) is one of the well-known approaches for data mining [23]. The patterns discovered using FIM are often in the form of association rules defined by a few critical feature items, which are extracted from a series of frequently occurring transactions [24]. Similarly, in network traffic analysis, each network flow can be treated as a transaction, with protocol-related and flow-level features modeled as items. Then, the pattern extraction of anomalous traffic flows is equivalent to mining the anomalous association rules from flow-level data. A recent work [25] proposed an FIM-based framework for detecting attacks specifically in distributed IoT networks. They mined the patterns to identify spatial and temporal correlations from aggregated alerts generated by home networks; these mined patterns were then fed to a supervised classifier to detect different stages of attacks. Coming to the traditional networks, Shone et al. [26] introduced a framework to extract the rules associated with anomalous traffic flows in the backbone networks. They filtered the network flows by means of a set of meta-data of network traffic, which are jointly contributed by multiple histogram-based anomaly detectors, and then extracted the rules of suspicious behaviors from filtered flows based on the Apriori algorithm [27]. However, these works still do not provide a solution to explain the anomalies or to attribute the anomalies to specific attack classes for generic networks (such as enterprises, homes, etc.). Thus, in this work, we explore a systematic way for extracting behavior patterns and identifying attacks, so as to comprehend and explain anomalies in network traffic.

III. OVERVIEW OF APEX

As APEX is based on FIM, we first give a brief background of FIM, followed by an overview of APEX. Finally, we present the threat model.

A. Frequent Itemset Mining (FIM)

In FIM, each field of a transaction is called an item, and a set of k items is called a k -itemset, where k is also referred to the length of the itemset. Note that the terms itemset and pattern are used interchangeably here. Each itemset is associated with an attribute called *support* which indicates the number of times the respective pattern is found in the transactions. An itemset is called a frequent itemset, if its support is beyond a given minimum support, θ ($0 \leq \theta \leq 1$). Note that, the support is sometimes used as an absolute value and in this case, it is referred to as support count (see Fig. 2).

Researchers have proposed different FIM algorithms such as Apriori [27], Charm [28], and FPMMax [29], that extract different sets of patterns. Apriori is useful in extracting a set of all possible frequent itemsets (FI); such a set is also called a “lattice”. Although a lattice provides a comprehensive overview of all the patterns, the number of patterns is very high, and more importantly, lower length itemsets¹ are usually

¹The length of an itemset is equal to the number of items in the itemset.

Fig. 1. Overview of the framework.

subsets² of higher length itemsets, and thus redundant. For example, $\langle\langle a, c \rangle\rangle$ and $\langle\langle a, b, d \rangle\rangle$ are two subsets of the itemset $\langle\langle a, b, c, d \rangle\rangle$. Alternatively, we can mine for closed frequent itemsets (CFI) and maximal frequent itemsets (MFI), both of which are subsets of FI. The itemsets in CFI do not have any superset with the same support, whereas the itemsets in MFI do not have any frequent superset. A detailed demonstration of three kind of itemsets in lattice diagram is presented in the Section I of the online supplementary file.

CFI and MFI can be mined with Charm and FPMMax algorithms, respectively. While MFI contains less number of itemsets compared to CFI, the length of the itemsets (the number of items) in MFI, which is related to the information content, is also higher. Therefore, in APEX, we mine for MFIs in both levels of mining. We present a comparison between the results of mining different sets later in Section V.

B. APEX Framework: An Overview

APEX aims to achieve two goals: i) to characterise anomalies detected, ii) unlike supervised models, be less dependent on data, while not compromising the detection capability. Fig. 1 presents the APEX framework, consisting of a training phase and a testing (or operational) phase. Both phases operate on anomalous traffic flows, which are obtained by employing one of the existing anomaly detection solutions. A practical approach is one that is not dependent on the availability of labels for both normal and anomalous flows. GEE [7], a recent proposal, is an apt candidate that uses an unsupervised VAE (variational autoencoder) model to detect anomalies in network traffic. In this work, we define a network flow as a set of packets with the same five-tuple of source and destination IP addresses, source and destination ports, and protocol. Two flows with the same five-tuple that are separated in time by a sufficiently long silent period are considered as separate flows.

APEX performs two levels of mining in the training phase. First, the anomalous traffic flows obtained as output from the anomaly detection solution, say GEE, are mined with a set of features with FIM to extract different patterns (Section IV-A).

In Fig. 2, we list down real example patterns mined from the anomalous flows detected by GEE on MAWI data (see Section V-A for details on the dataset used). The extracted patterns reflect different attack profiles coming from or related to different sources. Next, assuming the information about attack types are available, we aggregate patterns from the first level of mining according to attack types such as *Scan*, *Flood*, etc. We refer to these class-wise aggregated patterns as *fingerprints*. For example, the mined patterns corresponding to *Scan* in Fig. 2 are some of the fingerprints for that particular attack class.

Next, in order to have concise patterns representing each attack class, a second level of mining is carried out on the fingerprint patterns of respective attack classes (Section IV-B). Fig. 3 illustrates fingerprints (rows 1 to 10) and the patterns mined from them (rows I and II). Since there are many such patterns for each class, we identify the top m patterns as the *behavioral patterns* of each attack class and use them as identifiers in the testing phase. Examples of behavioral patterns of each class are shown in Table I.

In the testing phase, anomalous five-tuple traffic flows go through the first level of mining, and the extracted patterns are compared with behavioral patterns of different attack classes for possible identification of attacks present in the network traffic. As the mining may extract multiple patterns (that are associated with multiple attack types) from network flows, we propose and evaluate three majority-voting based matching schemes to identify the attack type originating from a source IP address (Section IV-C).

C. Threat model

In this work, we consider different kinds of network attacks that manifest as multiple flows. These include port scans, network scans, C&C communications, volumetric attacks such as floods and DDoS (distributed denial-of-service), brute-force login attempts, etc. In case of multi-flow attacks, the network flows of an attack might occur between two hosts (point-to-point attack) or across multiple hosts (e.g., one server being attacked by a botnet). Our work focuses on detection and characterising such attacks. On the other hand, single-flow attacks, such as SQL injection attack that uses just one network flow, are considered outside the scope of our work, since for a data-mining framework these attacks have very low support (equivalent to that of noise).

In the context of this work, we define four high-level attack classes—*Scan*, *Heavy*, *Flood*, and *Low-rate*—based on the availability of data—which are described in more details in Section V-A. It is important to note that the framework of APEX is generic, and can be used with other definitions of attack classes.

IV. ATTACK MINING AND IDENTIFICATION

A. First-Level Mining: Extracting Patterns

The anomalous traffic flows from a network anomaly detection form the input to APEX, specifically to the first level of mining (used in both training and testing phases). Specifically, APEX takes the top-ranked anomalous source IP addresses

²In FIM, itemset B is a subset of A, if all the items in B are contained in A and the length of A is higher than that of B.

Features of first level of mining														support count
	ID	src IP	dst IP	proto	src port	dst port	AIPL	FSP	FSB	flow duration	FSB-FSP	MPL	simplified flags	
Scan	1	197.131.116.159	*	6	35291	23	S	S	S	S	T	S	S_R	24
	2	139.98.16.223	*	6	*	22	S	S	S	S	T	S	S_RA	223
	3	78.106.181.65	*	6	*	445	S	S	S	S	T	S	S	2343
	4	36.119.49.26	*	17	50105	53413	S	S	S	S	T	S	_	9
	5	190.45.124.105	*	6	*	1433	S	S	S	S	T	S	S	30
Heavy	6	133.26.137.232	61.164.52.47	17	4500	33679	L	M	L	*	*	S	_	48
	7	173.195.52.59	163.94.126.61	6	443	58669	L	L	L	*	H	L	A_PA	44
	8	74.146.223.54	202.4.32.242	6	443	*	L	L	L	M	H	L	A_PA	55
	9	203.162.154.80	218.62.66.127	6	80	59993	L	L	L	L	H	L	A_PA	60
	10	203.162.157.77	*	6	*	*	L	L	L	*	H	L	*	11

Fig. 2. Example patterns from the first level of mining grouped into attack types - fingerprints. The wildcard “*” represents a collection of multiple values. Note, these patterns are mined from unidirectional traffic flows in MAWI data.

from an anomaly detection solution, and performs mining on their corresponding five-tuple flows. For mining these five-tuple anomalous network flows, we define a set of features (that form items in FIM) useful in detecting various attacks.

IP Destination³: This feature is useful in identifying the direction of the attack. If the item is mined as a wildcard (represented by a “*”), then the source is connecting to multiple destinations, a typical behavior of network scans (rows 1 to 5 in Fig. 2 are examples). On the other hand, if the destination IP address is unique, it means the flow is a one-to-one connection; e.g., row 6 in Fig. 2 shows a flow with source 133.26.137.232 connecting to destination 61.164.52.47. Note that instead of specific IP addresses, we can consider subnets, and in that case, attacks pertaining to specific subnet will be detected. We demonstrate a pattern from Flood-like flows related to subnets later in the Section V-D.

Protocol: Different protocols (e.g., TCP and UDP at transport layer) are numbered and used as categorical attributes.

Source Port and Destination Port: Similar to protocol, we use port numbers as categorical features since they indicate the applications in use. If a port number is repeated in several flows, it will be extracted as a frequent item for those set of flows. Some common ports, for example 80 and 443 used for HTTP(S), display such a behavior; this is illustrated in Fig. 2. **Average IP Packet Length (AIPL):** This is computed per five-tuple flow as the ratio of the sum of the IP packet length (in bytes) to the total number of IP packets in the flow. This feature is further categorized into three bins as small (S), medium (M), and large (L).

Flow Size in Packets (FSP): FSP is the total packet count in a flow, and is divided into three bins. For Heavy flows, due to their intrinsic nature of large sizes, FSP would often be large. On the other hand, Scans and Low-rate attacks manifest as smaller size flows. Fig. 2 illustrates this contrast between Scan and Heavy.

Flow Size in Bytes (FSB): FSB is the total size of a flow in bytes, and is also categorized into three bins. This feature is similar to FSP, but could capture subtle differences. For instance, while both Scan and Low-rate attacks usually have a small number of packets in their five-tuples, Low-rate flows may have larger packets due to the presence of application data (e.g., SSH brute force attempt would contain key-exchange messages, user authentication data, etc.).

Flow Duration: The duration of the flow is also divided into three discrete bins. In short bursty flows such as Floods, the flow duration is likely to be short, whereas in Low-rate attacks the flows could prevail for a long period of time than Scans. Going back to the SSH brute-force example, flows may have a longer duration depending on, say, the number of user authentication attempts carried out by the attacker [30], and similar behavior is possible for other forms of dictionary attacks as well.

FSB-FSP: This is a two-dimensional feature with two categorical bins. It jointly describes the correlation between FSB and FSP through categorizing the ratio of FSB to FSP into different bins.

Majority Packet Length (MPL): MPL represents the major category (in terms of packet length) of the packets with a payload in a flow. We use three categories for this feature.

Simplified Flag Sequence: TCP flag sequence can reveal characteristics of attacks. For example, a SYN Scan has a number of TCP packets with SYN flag set. Similarly, interactive applications tend to set the TCP PUSH flag, and therefore Low-rate attacks on certain applications may show differentiation on flag sequences. Representing the sequence of flags present in all packets of a flow is not possible (due to the long lengths of such sequences). Therefore, with this feature we capture a simplified version of the complete flag sequence where different flags are placed in a pre-defined order regardless of the occurrence and position they are in the original flag sequence. If the complete flag sequence is S_A_PA_A_PA_A_PA_A_FA_A, the corresponding simplified version

³Mining in APEX is performed on the flows from the same source IP address. Thus IP source address will always be extracted as a frequent item.

would be $S_A_PA_FA^4$.

In addition to the aforementioned features, we also consider two more features based on DNS and TLS traffic. We provide a brief discussion in Section II-A of the online supplementary materials regarding these two features. However, these two features are not included in the pattern examples (extracted from the real traffic) or in the experiments later, as the MAWI dataset does not provide the relevant information.

Boundaries for feature categories. The margins of the bins of the above categorical features are estimated based on both domain knowledge and statistical analysis of attack traffic. Later in Section V, we provide the boundary values used for our experiments.

Mining. Once the features are extracted, anomalous five-tuple flows represented using the above set of features are mined with FIM to extract patterns. As mentioned earlier, we mine for MFI using FPMMax algorithm [29] as MFI tends to provide a minimum set of patterns with higher lengths (more frequent items). As the first-level mining is performed without using the attack labels, this part of the training is unsupervised. Note that here we only mine flows originating from a single source IP address at a time; this is quite well-aligned with anomaly detection solutions that often predict the anomalous sources in a given time-slot since aggregation of flows at source level gives more information for better classification [6], [7].

B. Second-Level Mining: Extracting Behaviors

The second-level mining is executed only in the training phase. As the first step, we aggregate the itemsets or patterns obtained as output from the first-level mining into different attack types. To do so, we assume the availability of labeled attack traffic. Note that, labeled attack flows are regularly detected by different solutions as well as with the help of analysts; therefore, obtaining this small labelled set is not as difficult as obtaining labels on all flows (which is required for supervised learning). We refer to the grouped and labeled attack patterns as *fingerprints* of attack classes. A few real fingerprint patterns of Scan and Heavy attack classes are shown in Fig. 2.

Motivation. The number of fingerprints for a given class can be quite large; even for one type of attack, there can be numerous fingerprint patterns from the first level of mining. For example, consider a set of bots scanning for vulnerable hosts in a subnet on a specific port, say 445. The first-level mining may generate a pattern for each of the bot, resulting in many patterns. However, another round of mining on these patterns could result in a few patterns that represent the attack behavior more concisely. Subsequently, the task with the second level of mining is to extract the behaviors from these fingerprint patterns. Fig. 3 provides examples of second-level mining with respect to Flood and Low-rate attack classes.

Features for second-level mining. A closer look at the patterns of different attack classes obtained from first level of mining (refer to rows 1 to 10 in Fig. 2 and Fig. 3) shows distinct behaviors in some features. For instance, in Scans

(rows 1 to 5 in Fig. 2), FSB, FSP, and flow duration tend to be small, whereas in Heavy (rows 6 to 10 in Fig. 2), average IP packet length and FSB tend to be large in most patterns. While Scan and Low-rate attacks (rows 6 to 10 in Fig. 3) appear similar and thus challenging to differentiate, they show differences in the flag sequences. Similarly, Heavy and Flood display similar behaviors, but items such as flow duration tend to be different in these attacks.

In addition to the features in the first level of mining, we include support of the extracted patterns as a feature for the second level of mining. In mining, support indicates the number of times a captured pattern is repeated in the transaction database. Based on the dominance of the pattern and correspondingly the attack, the support will vary. For instance, a flooding attack is a burst of flows within a short time period and thus, in the corresponding captured pattern, the support will be high. In the second-level mining, we categorize the support into four bins (see Table III). Additionally, we include the support as a numerical feature as well. In some attacks, the same number of flows can be seen in the network traces (e.g., a malware might scan 10 different ports on one target, before moving to another target), and the support as a numerical feature can help to detect those repetitive patterns.

If the destination IP address is a unique address (not a wildcard), we mark it as “v” before feeding to the second level of mining in order to capture flows with unique destinations. Rows 1, 3-5, and 6-10 of Fig. 3 illustrate this.

Mining. The main objective of the second level of mining is to extract these repeated items in the patterns of the first level of mining. As explained before, these key items can be used as identifiers in the detection of attack types. Similar to the first level, we mine for MFI in the second level for the same reason that MFI generates more informative patterns of higher length (than CFI). In Fig. 3, at the end of fingerprint patterns of Flood and Low-rate attack classes (i.e., in rows I and II), we depict the results from the second level of mining.

Out of the patterns from the second level of mining (which describe the frequent items in the attack patterns), we select the top m patterns as the *identifiers* that capture the behavior of the corresponding attack type. We refer to them as “*behavioral patterns*” and use them in the testing process as identifiers. Table I provides examples of behavioral patterns with respect to each attack class.

C. Attack Identification

Attack identification is carried out in the testing phase. First, similar to training, anomalous five-tuple flows are obtained from a network anomaly detection solution. From these anomalous flows, the set of features used for the first level of mining are extracted. Subsequently, these transactions are mined for MFI (first level of mining) with an identical minimum support to training. The resulting *testing patterns* are then cross-checked with behavioral patterns of each class with the following matching schemes to identify the attack class. Specifically, in the matching schemes defined below, we evaluate whether the items in behavioral patterns (obtained in the training phase) are contained in a testing pattern as the

⁴This is for one direction of a flow; clearly, the bidirectional sequence would be different as both SYN (S) and SYN+ACK (S_A) would be captured.

Features of second level of mining														
ID	src IP	dst IP	proto	src port	dst port	AIPL	FSP	FSB	flow duration	FSB - FSP	MPL	simplified flags	support count	
1	74.146.213.79	163.94.126.46 (V)	17	443	*	L	L	L	S	H	S	_	11 (S)	Flood
2	31.53.61.250	*	6	*	*	L	M	L	*	H	L	SA_A_PA	73 (S)	
3	74.146.213.79	163.94.53.57 (V)	6	443	59289	L	*	L	S	H	L	*	10 (S)	
4	61.184.33.171	203.162.154.90 (V)	6	*	443	M	S	S	S	T	S	PA	8135 (L)	
5	74.146.218.17	202.4.32.242 (V)	6	443	*	L	L	L	M	H	L	SA_A_RA	27 (S)	
I		(V)	6	443	*	L		L	S	H	L		(S)	Low-rate
6	202.36.147.165	163.94.238.104 (V)	6	8081	*	M	S	S	S	T	S	SA_A_FA	19 (S)	
7	163.94.2.187	52.69.157.196 (V)	6	*	443	M	S	S	L	T	S	S_A_PA_FA	152 (M)	
8	202.9.77.196	204.178.59.42 (V)	6	*	443	M	M	S	L	T	S	S_A_PA_FA	597 (M)	
9	47.85.188.7	203.162.144.17 (V)	17	*	53	M	M	S	S	T	S	_	587 (M)	
10	34.203.201.110	163.210.205.51 (V)	6	80	*	M	S	S	L	T	S	S_A_PA_FA	226 (M)	
II		(V)	6	*		M	S	S	L	T	S	S_A_PA_FA	(M)	

Fig. 3. Rows 1 to 10 are fingerprint patterns mined from first level of mining. Rows I and II (marked on the left) are behavioral patterns mined by second-level mining from fingerprint patterns. “(V)” denotes unique IP addresses are captured as a frequent item, and “_” that there is no frequent item. Note, these patterns are mined from unidirectional traffic flows in MAWI data.

former (which is from the second level of mining) usually have lesser number of items compared to the latter (which is from the first level of mining). For our description below, we use the behavioral patterns shown in Table I as examples.

TABLE I
EXAMPLE BEHAVIORAL PATTERNS. THE ORDER OF FEATURES IN THESE PATTERNS IS THE SAME AS THE FEATURES LISTED IN FIG. 3.

Attack	#	Behavioral Patterns ($m=3$)
Scan	1a	$\langle\langle_*, 6, *, _, S, S, S, S, T, S, _, _ \rangle\rangle$
	1b	$\langle\langle_*, 6, _, 23, S, S, S, _, T, S, _, _ \rangle\rangle$
	1c	$\langle\langle_*, 6, _, _, S, S, S, S, T, S, S, _ \rangle\rangle$
Heavy	2a	$\langle\langle_ (V), 6, _, _, L, L, *, H, L, A_PA, _ \rangle\rangle$
	2b	$\langle\langle_*, 6, _, _, L, _, L, _, H, L, _, _ \rangle\rangle$
	2c	$\langle\langle_ (V), 6, 443, _, L, *, L, _, H, L, _, _ \rangle\rangle$
Flood	3a	$\langle\langle_ (V), 6, 443, *, L, _, L, S, H, L, _, _ \rangle\rangle$
	3b	$\langle\langle_*, 6, 443, *, L, _, L, _, H, L, A, _ \rangle\rangle$
	3c	$\langle\langle_*, _, _, _, L, *, *, S, T, S, _, _ \rangle\rangle$
Low-rate	4a	$\langle\langle_ (V), 6, *, _, M, S, S, L, T, S, S_A_PA_FA, _ \rangle\rangle$
	4b	$\langle\langle_ (V), _, _, *, S, S, *, S, T, _, _ \rangle\rangle$
	4c	$\langle\langle_*, 6, _, _, M, S, S, *, T, S, _, _ \rangle\rangle$

Exact Match: If all items in a behavioral pattern of a class match with the items in the testing pattern, then it is considered a valid match under this scheme. This matching is executed across all the behavioral patterns from the different attack classes. For instance, if $\langle\langle 83.48.50.62, *, 6, *, 1900, S, S, S, S, T, S, S, 35 (S) \rangle\rangle$ (the penultimate item is Simplified Flag Sequence) is the testing pattern, then only the first (1a) and third (1c) behavioral patterns of Scan exactly matches with the testing pattern — all the items in the first and third behavioral patterns of Scan are contained in the testing pattern.

Best Match: In this scheme, the behavioral pattern of each class that matches with most of the items in the testing pattern is considered, and a score is calculated as a percentage of total items matched in the

respective behavioral pattern. Consider a testing pattern, $\langle\langle 74.253.96.81, 163.221.98.150, 6, 443, 51672, L, L, L, M, H, L, A_PA, 45 (S) \rangle\rangle$. If we verify this pattern against the behavioral patterns of each class in Table I, best matching patterns are 1b for Scan, 2a for Heavy, 3a for Flood and 4a for Low-rate. The corresponding matching scores for best matching patterns are; Scan : 1/8 (only the protocol TCP matches), Heavy : 8/9 (there is a match for all items in 2a, except for the flow duration), Flood : 7/9 (all items in 3a match except for the destination port and flow duration), Low-rate : 2/10 (only destination IP address and protocol match in 4a). Hence, the testing pattern will be classified as Heavy.

Overall Match: Here, matching percentages of all the behavioral patterns with the testing pattern are evaluated and the final score is taken as an average: $\sum_{i \in m} \left(\frac{\# \text{ matched items}}{\# \text{ items in the behavioral pattern } i} \right) / m$. If the testing pattern is: $\langle\langle 163.194.93.97, 192.129.24.183, 6, *, 443, M, S, S, L, T, S, S_A_PA_FA, 122 (M) \rangle\rangle$, then the scores after verifying with all the behavioral patterns in Table I are:

- Scan: $\frac{(\frac{7}{9}) + (\frac{6}{8}) + (\frac{6}{9})}{3} = 0.73$;
- Heavy: $\frac{(\frac{2}{9}) + (\frac{1}{5}) + (\frac{2}{8})}{3} = 0.22$;
- Flood: $\frac{(\frac{2}{9}) + (\frac{3}{8}) + (\frac{3}{8})}{3} = 0.24$;
- Low-rate: $\frac{(\frac{10}{10}) + (\frac{4}{7}) + (\frac{6}{8})}{3} = 0.77$.

Finally, with an overall score of 0.77, Low-rate will be selected as the attack class of the above testing pattern.

After all the patterns corresponding to a source IP address are evaluated, the source is classified to an attack class based on majority vote — to the attack class to which the majority of patterns are classified. In case the voting is equal for two or more attack classes, the number of matching frequent items in the corresponding behavioral patterns are considered; i.e., the attack class of the behavioral pattern(s) with more matched

Algorithm 1 Extraction of level one patterns

-
- 1: $\text{Flows} \leftarrow$ Apply unsupervised anomaly detection to obtain anomalous five-tuple flows
 - 2: $\text{Grouped Flows} \leftarrow$ Group Flows based on source IP addresses
 - 3: $\text{Patterns} \leftarrow$ Apply First-level mining using FPMMax on Grouped Flows (Section IV-A)
 - 4: **return** Patterns
-

Algorithm 2 Training phase of APEX

-
- 1: $\text{Patterns}_1 \leftarrow$ Extract level one patterns (Algorithm 1)
 - 2: $\text{Fingerprints} \leftarrow$ Group Patterns_1 based on attack type
 - 3: Let $\mathcal{C}$ be the set of attack classes
 - 4: **for each** $c \in \mathcal{C}$ **do**
 - 5: $\text{Patterns}_2 \leftarrow$ Apply Second-level mining using FPMMax on Fingerprints of attack class c
 - 6: $\mathcal{B}_c \leftarrow$ Select top m patterns from Patterns_2 (Section IV-B)
 - 7: **end for**
 - 8: **return** $\mathcal{B}$ $\triangleright$ return behavioral patterns
-

frequent items will be selected. Note that the considered behavioral patterns for this final decision depend on the matching scheme; in exact match, all the matching behavioral patterns are considered; in best match, the top scoring behavioral pattern is considered; in overall match, all the behavioral patterns are considered.

D. Summary of APEX

We summarize APEX from the perspective of training and testing operations. As described before, both phases perform the first-level mining on anomalous network flows. These common steps are listed in Algorithm 1. The training phase mines and returns behavioral patterns, as shown in Algorithm 2. The testing phase, Algorithm 3, uses the behavioral patterns for matching against the mined pattern from the test data.

Algorithm 3 Testing phase of APEX

-
- 1: $\text{Patterns}_1 \leftarrow$ Extract level one patterns (Algorithm 1)
 - 2: Let $\mathcal{B}$ be the behavioral patterns from training phase
 - 3: $\text{Results} \leftarrow$ Apply matching Scheme $\mathcal{M}$ on each pattern $P \in \text{Patterns}_1$, across m behavioral patterns in $\mathcal{B}_c, (\forall \mathcal{B}_c \in \mathcal{B})$ (Section IV-C)
 - 4: **return** Results
-

V. PERFORMANCE EVALUATION*A. Network traffic dataset from MAWI archive*

For our experiments here, we use packet captures (pcaps without payloads) publicly available at the MAWI traffic archive [9]. The dataset we use consists of seven randomly selected days from Sep. 2018 to Dec. 2018; there are more than 1 billion packets and 60 million flows in total. We first perform data cleaning to remove source IP addresses that have very few packets (< 20). Subsequently, in order to focus on IP addresses

that are more likely to be anomalous, we apply a VAE-based unsupervised anomaly detector, specifically GEE [7]. In this work, we select around 640 of the top source IP addresses that are predicted as anomalous by GEE based on their ranking of the anomaly scores. Subsequently, we analyze five-tuple flows (silence period is set to 5 seconds, which is used to separate flows with same five-tuples) from these reduced set of source IP addresses, and manually label them into four attack classes as defined below.

From the pcap files corresponding to the different source IP addresses, labels were assigned based on manual and semi-automated analysis for the given source IP addresses. However, in the manual labeling process, the traffic characteristic of some source IP addresses are determined to be not malicious even though they are given high anomaly scores by the VAE model. We label these IP addresses as *Normal* and do not include them in the evaluations.

Attack Classes. The following high-level attack classes are defined based on the dataset used for experiments here.

Scan: Host and network scanning are integral components of any malware and are used to look for potential vulnerabilities. While the basic idea of scanning is obvious, a large variety of scanning strategies exist and it is non-trivial to automatically identify scanning traffic from a large network traffic trace.

Heavy: This is a traffic class for high volume flows. In order to differentiate different high volume source IP addresses, we identify a source (IP address) as *Heavy* only if there is at least one five-tuple flow with sufficiently high volume with a certain threshold. In our experiments, the threshold is set to flow size measured in bytes that is greater than 99% of all flows in the original data, within the interval of interest.

Flood: This is also a traffic class for high volume flows. The difference between *Flood* and *Heavy* is that instead of one (or a few) elephant flow(s), *Flood* is characterised by having a large number of flows to one or different destinations. In aggregation, the flows are voluminous.

Low-rate: These are malicious flows characterized by a large count, and each flow only transfers a small number of packets at relatively low rate. An example of such *Low-rate* attack is slowloris [31]. C&C communications as well as brute-force login attempts also fall under this class.

The attack class distribution is given in Table II. We publish the labels and information of anomalous source IP addresses⁵.

TABLE II
DISTRIBUTION OF ATTACK TRAFFIC FROM SEVEN DAYS OF 2018

Date (mm-dd)	No. of five-tuple flows / No. of source IP addresses			
	Flood	Heavy	Low-rate	Scan
09-01	56K / 16	55K / 25	8K / 18	412K / 37
09-02	75K / 15	81K / 18	121K / 16	268K / 32
09-06	300K / 5	64K / 11	118K / 6	502K / 32
09-07	292K / 10	35K / 12	70K / 6	741K / 54
10-06	146K / 21	26K / 13	147K / 11	220K / 41
11-07	109K / 20	44K / 18	258K / 9	294K / 27
12-18	665K / 21	106K / 11	60K / 15	671K / 58
Total	1646K / 108	414K / 108	861K / 81	3111K / 281

The boundary values for the categorical features we use in our experiments are listed in Table III. They are derived based

⁵<https://github.com/Oshirobeta/apex-ip-labels>.

on our understanding of the attack classes and the practical feature distributions based on the MAWI traffic. The letters T, S, M, L, and H, denote Tiny, Small, Medium, Large, and High, respectively. We provide a demonstration on derivation of the boundary values for *AIPL*, *Flow duration* and *FSB-FSP* in Section II-B of the online supplementary materials.

TABLE III
BOUNDARIES OF CATEGORICAL FEATURES.

Feature	Boundary
AIPL (byte)	$S \leq 70 < M \leq 500 < L$
FSP (count)	$S \leq 10 < M \leq 100 < L$
FSB (byte)	$S \leq 3000 < M \leq 10000 < L$
Flow Duration (s)	$S \leq 1 < M \leq 5 < L$
FSB-FSP (byte/count)	$T \leq 661.76 < H$
MPL (byte)	$80 < S \leq 500 < M \leq 1000 < L$
Support (count)	$T \leq 10 < S \leq 100 < M \leq 1000 < L$

B. Evaluation Metrics

For an attack class c , let TP_c , TN_c , FP_c , and FN_c denote the counts of true positives, true negatives, false positives, and false negatives, respectively. If $\mathcal{C}$ denotes the set of all classes, then the overall Precision, Recall and F1-Score are defined:

$$\text{Precision} = \frac{\sum_{c \in \mathcal{C}} TP_c}{\sum_{c \in \mathcal{C}} TP_c + \sum_{c \in \mathcal{C}} FP_c}; \text{Recall} = \frac{\sum_{c \in \mathcal{C}} TP_c}{\sum_{c \in \mathcal{C}} TP_c + \sum_{c \in \mathcal{C}} FN_c};$$

$$\text{F1-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (1)$$

Since data in the attack classes can be imbalanced, we also use the Weighted F1-Score, which is based on the F1-Score of each class. Given N_c is the number of data points belonging to attack class c , the F1-Score_c of class c is:

$$\text{F1-Score}_c = \frac{2 * TP_c}{2 * TP_c + FP_c + FN_c}, \forall c \in \mathcal{C}; \quad \text{and,}$$

$$\text{Weighted F1-Score} = \frac{\sum_{c \in \mathcal{C}} N_c * \text{F1-Score}_c}{\sum_{c \in \mathcal{C}} N_c}. \quad (2)$$

For our evaluations below, we analyze based on the overall F1-Score (Eq. 1) and Weighted F1-Score (Eq. 2).

C. Results

We carry out four sets of experiments to investigate the best pattern mining and matching scheme of APEX, and also to compare the detection performance of APEX with supervised machine learning models. m , the number of behavioral patterns to compare, is the most important parameter of the matching scheme for APEX. Except in *Experiment B* where we analyse the effect of m , in all other experiment scenarios m is set to 7. The support for mining, in both the first level and second level, is set to 0.1 across all experiments.

1) *Experiment A: FI vs CFI vs MFI*: In this experiment, we analyse which of the three different sets of patterns — FI, CFI and MFI (refer Section III-A) — are effective in identifying attack patterns. We use three well-known mining algorithms Apriori [27], Charm [28], and FPMMax [29] to mine for FI, CFI, and MFI, respectively. APEX is trained using the 10-06 dataset, while the testing is conducted on the 11-07 dataset. Seven behavioral patterns ($m = 7$) are selected for each attack class at the second level of mining.

Fig. 4. FI vs CFI vs MFI

The results are presented in Fig. 4. Clearly, MFI achieves the best performance on all the metrics. The fact that patterns in MFI are well descriptive (more items in patterns) helps to outperform FI and CFI. While FI extracts all the patterns including lower length patterns, CFI extracts the patterns with higher supports that could also be in lower lengths. With MFI, the extraction of higher-length patterns at the first level of mining helps to have behavioral patterns with more frequent items at the second level of mining; this eventually leads to a better classification of the attack classes. As MFI yields the best performance, we mine for MFI in both levels of mining in the remaining experiments.

2) *Experiment B: Analysis of Matching Schemes and the impact of m* : Next, we compare the effect of m in the performance of the different matching schemes. Recall, m denotes the number of behavioral patterns mined in the training phase; these behavioral patterns are matched against, by the mined patterns in the testing (operational) phase. As generating these behavioral patterns requires labeled attack traffic and hence human assistance, high values of m is not preferred. Hence, we experiment for values of m set to 3, 5, 7, 9, and 11. The 09-01 dataset is used in the training phase, and the 09-06 dataset is used for the testing phase. The experiment is repeated for all values of m , and we compare the three matching schemes (Section IV-C) in their ability to identify the different attack classes. The corresponding results are shown in Fig. 5. In Exact Match, when m increases, the number of behavioral patterns available to identify the attack class also increases, and thus the probability of getting an accurate match improves. This is evident from Fig. 5, where the overall F1-Score of Exact Match increase from ~ 0.3 to over 0.6 with increasing m . However, as the number of behavioral patterns m increases, it becomes harder to characterize attacks; besides, extracting more number of behavioral patterns would require more number of labeled attack flows. The Best Match and Overall Match consistently score over 0.5 for different values of m . On average, Best Match shows better performance than Overall Match — in terms of the overall F1-score, Best Match averages to ~ 0.6 , whereas Overall Match averages to 0.57. Henceforth, we use Best Match in the remaining experiments.

3) *Comparison with supervised approaches*: We compare APEX with several supervised machine learning models that are widely adopted for classifications: Random Forest, k-Nearest Neighbor (k-NN) and Naive Bayes. Observe that, supervised models require labels for all traffic flows, whereas APEX requires labels of only attack flows which can be

Fig. 5. Performance with different m values and matching schemes.

obtained by analyzing the top-ranked anomalies of an anomaly detection model. Since supervised models are expected to outperform unsupervised/semi-supervised models, with this set of experiments our goal is to analyse how close is APEX to supervised models in identifying attack classes.

To have a fair comparison of the supervised learning models and APEX, we engineer the features as follows. The supervised models are trained and tested on five-tuple flows, where the features of *Source IP address*, *Destination IP address*, *Source Port*, *Destination Port* and *Protocol* (used in APEX) become the identifiers to extract the flows. The numerical features such as *AIPL*, *FSB*, *FSP* and *Flow Duration* are also features in machine learning models. Two categorical features are engineered to accommodate in the machine learning models. The *Simplified Flag Sequence*, is transformed to the basic flag counts in all the IP packets within a flow. The other categorical feature, *Majority Packet Length*, is converted to three-dimensional binary feature vector through one-hot encoding, i.e., the categorical values of S, M, L are converted to [0,0,1], [0,1,0] and [1,0,0] respectively. Since with machine learning models the classification is performed at the flow level, a majority voting based scheme is employed to assign the attack class originating from the corresponding sources. As mentioned before, for APEX, the minimum support is set to 0.1; and m is set to 7 as it encompasses a manageable number of behavioral patterns with a reasonable performance.

a) *Experiment C: Effect of training dataset size on performance*: We usually expect learning algorithms to perform better with increasing training dataset; in this experiment we evaluate APEX and the supervised machine learning algorithms with two sets of network traces. First set uses network traces of four days, i.e., 09-01, 09-06, 10-06, 11-07, for training, and the remaining three days to test. The second set uses five days, i.e., 09-01, 09-02, 09-06, 10-06, 11-07, to train the models and the remaining two days to test.

The results are shown in Fig. 6. Observe that, when the size of training data increases, different from k-NN and Naive Bayes, the performance of both Random Forest and APEX improves. Importantly, with more training data, the extent of APEX's performance improvement is greater than that of Random Forest; and in fact, with 5:2 ratio APEX outperforms both k-NN and Naive Bayes. Finally, the results show that, APEX achieves comparable performance to the supervised machine learning models, given that APEX is semi-supervised.

b) *Experiment D: Performance on entire dataset*: In this experiment, we merge all the source IP addresses in seven

Fig. 6. Performance with different dataset combinations.

Fig. 7. Performance on global data.

days of network traces and split them into training and testing at the ratio of 70%:30%, while also keeping the same training: testing ratio for each class. As the result shows in Fig. 7, when a comprehensive training dataset is used, APEX is able to achieve as high a performance as Random Forest; APEX also outperforms the other two supervised models. Besides being able to identify attack classes, APEX is also able to provide behavioral patterns of attacks. These patterns provide useful contexts to analysts; in addition, analysts can also use these patterns to quickly form rules to take mitigation actions.

D. Attack Patterns: A Discussion

Subnet Pattern. We use a topological diagram (Fig.8) to demonstrate a Flood type attack observed from the subnet-related patterns (based on the first level of mining) in the MAWI dataset. In this instance, patterns are formed between multiple sources and a single destination subnet, 163.94.0.0/16, over the ports of 80 (HTTP) and 443 (HTTPS). Note that both the average packet length (AIPL) and majority packet length (MPL) are in the “L” (large) category, and the support of these patterns goes beyond 10,000. Hence,

Fig. 8. Topological Flood-like pattern.

these patterns are most likely from a volumetric attack such as HTTP Flood.

Observe that, the traffic flows are from web servers (ports 443 and 80) to the subnet 163.94.0.0/16; that is, they are responding to the connections initiated by this particular subnet. Therefore, the subnet 163.94.0.0/16 is the anomalous entity in this example. We highlight that, the datasets do not always contain bi-directional flows, and therefore no patterns from the subnet 163.94.0.0/16 to the web servers were observed. In addition, we provide another subnet-level pattern of *Scan* in Section III of the online supplementary materials.

Pattern distribution. We present an example using Sankey diagram⁶ to visually demonstrate the feature distribution of *Scan* attack patterns (Fig. 9) from the MAWI traffic. The patterns are derived from the first level of mining as elaborated in Section IV-A.

In the Sankey diagram, a pattern is viewed as a flow connecting a sequence of vertical bars of feature values from the left to the right. The left-most bar indicates the attack class, the rest of the columns are grouped based on features, and each bar representing a categorical feature value. The length of the bars are different based on the fraction of the five-tuple flows (corresponding to patterns) passing through them.

Fig. 9. Attack patterns of *Scan*

⁶Protocol, port numbers and support features are excluded from the Sankey diagram for better visibility.

In other words, it indicates how many flows attribute to the pattern that possesses such a specific feature value.

The pattern distributions presented in Sankey diagrams are useful in explaining results. In the case of *Scan* (Fig. 9), most feature values in the patterns are unique; the *AIPL*, *FSB*, *FSP*, *Flow duration* and *MPL* are in “S” (small) category for most patterns, and SYN flag (the last bar in the figure) is also present in majority of the extracted patterns. These unique feature values help to generate clear behavioral patterns in the second level of mining. For instance, consider the top behavioral patterns for *Scan* in Experiment D (Section V-C), which is specifically $\langle _ , _ , _ , _ , _ , _ , _ , _ , _ , _ \rangle$ (the order of features is the same as in Fig. 3 and “_” indicates that there is no frequent item). Note that the above behavioral pattern of *Scan* has many unique items. This observation in fact aligns with the Sankey diagram of *Scan* attack class (in Fig. 9) where the feature values are concentrated. In addition, we provide further demonstrations regarding the pattern distributions of the other three attack classes in the Section IV of the online supplementary file.

VI. CONCLUSION

In this work, we proposed a data mining framework APEX, to characterize network anomalies and classify them into different attack classes. APEX employed two levels of mining, where the first level extracted attack patterns from anomalous network flows, and the second level extracted a more concise representation of attack patterns called behavioral patterns which are then used in the identification of attacks. The experiments conducted on real network packet captures show that APEX achieves similar performance as the best performing supervised model, while providing a set of concise patterns matching attack classes as output. The patterns thus generated (and can be visualized) gives an analyst more context to the detected anomalies to carry out further investigations.

REFERENCES

- [1] RiskIQ, “Analysis of An Attack Surface: Five Ways Hackers Are Targeting Organizations,” RiskIQ, Inc, Tech. Rep., 2020. [Online]. Available: <https://www.riskiq.com/resources/research/analysis-of-an-attack-surface/>
- [2] MITRE, “MITRE ATT&CK,” 2021. [Online]. Available: <https://attack.mitre.org/>
- [3] L. Csikor, D. M. Divakaran, M. S. Kang, A. Korosi, B. Sonkoly, D. Haja, D. Pezaros, S. Schmid, and G. Rétvári, “Tuple Space Explosion: A Denial-of-Service Attack Against a Software Packet Classifier,” in *ACM CoNEXT*, Dec. 2019.
- [4] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, “Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis,” in *Proc. ACSAC*, 2012, p. 129–138.
- [5] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, “Who Are You? A Statistical Approach to Measuring User Authenticity,” in *Proc. NDSS*, 2016, pp. 1–15.
- [6] J. Kim, A. Sim, J. Kim, K. Wu, and J. Hahm, “Improving Botnet Detection with Recurrent Neural Network and Transfer Learning,” *arXiv:2104.12602*, pp. 1–13, 2021.
- [7] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, “GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection,” in *Proc. IEEE CNS*, 2019, pp. 91–99.
- [8] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A Deep Learning Approach to Network Intrusion Detection,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.
- [9] MAWI, “MAWI Traffic Archive,” 2018. [Online]. Available: <https://mawi.wide.ad.jp/mawi/>

- [10] Y. Gu, A. McCallum, and D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation," in *Proc. ACM IMC*, 2005, pp. 1–6.
- [11] F. Simmross-Wattenberg, J. I. Asensio-Perez, P. Casaseca-de-la Higuera, M. Martin-Fernandez, I. A. Dimitriadis, and C. Alberola-Lopez, "Anomaly Detection in Network Traffic Based on Statistical Inference and α -stable Modeling," *IEEE Trans. Dependable Secur. Comput.*, vol. 8, no. 4, p. 494–509, 2011.
- [12] I. Nevat, D. M. Divakaran, S. G. Nagarajan, P. Zhang, L. Su, L. L. Ko, and V. L. L. Thing, "Anomaly Detection and Attribution in Networks With Temporally Correlated Traffic," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 131–144, 2018.
- [13] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Proc. IEEE S&P*, 2010, pp. 305–316.
- [14] S. T. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing Darts in the Dark? Detecting Bots with Limited Data using Neural Data Augmentation," in *Proc. IEEE S&P*, 2020, pp. 1190–1206.
- [15] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Comput. Surv.*, vol. 31, no. 3, p. 264–323, 1999.
- [16] W. Lee and D. Xiang, "Information-theoretic Measures for Anomaly Detection," in *Proc. IEEE S&P*, 2001, pp. 130–143.
- [17] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-Scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks," in *Proc. ACSAC*, 2013, p. 199–208.
- [18] D. E. Rumelhart and J. L. McClelland, *Learning Internal Representations by Error Propagation*. MIT Press, 1986, vol. 1, pp. 318–362.
- [19] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proc. ICLR*, 2014, pp. 1–14.
- [20] M. Nicolau, J. McDermott *et al.*, "Learning Neural Representations for Network Anomaly Detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, 2018.
- [21] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proc. KDD*, 2016, p. 1135–1144.
- [22] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, "Explaining Anomalies Detected by Autoencoders using SHAP," *arXiv:1903.02407*, pp. 1–37, 2019.
- [23] R. Agrawal, T. Imieliński, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," in *Proc. ACM SIGMOD*, 1993, p. 207–216.
- [24] S. Moens, E. Aksehirli, and B. Goethals, "Frequent Itemset Mining for Big Data," in *Proc. IEEE BigData*, 2013, pp. 111–118.
- [25] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "ADEPT: Detection and Identification of Correlated Attack Stages in IoT Networks," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6591–6607, 2021.
- [26] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly Extraction in Backbone Networks using Association Rules," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1788–1799, 2012.
- [27] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proc. VLDB*, 1994, p. 487–499.
- [28] M. J. Zaki and C.-J. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," in *Proc. SIAM SDM*, 2002, pp. 457–473.
- [29] G. Grahne and J. Zhu, "High Performance Mining of Maximal Frequent Itemsets," in *Proc. SIAM HPDM Workshop*, vol. 16, 2003, p. 34.
- [30] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "SSH Compromise Detection Using NetFlow/IPFIX," *ACM SIGMOD*, vol. 44, no. 5, p. 20–26, 2014.
- [31] E. Cambiaso, G. Papaleo, and M. Aiello, "Taxonomy of Slow DoS Attacks to Web Applications," in *Proc. SNDS*, Berlin, Heidelberg, 2012, pp. 195–204.