

AP2Vec: An Unsupervised Approach for BGP Hijacking Detection

Tal Shapira^{1b}, *Graduate Student Member, IEEE*, and Yuval Shavitt^{1b}, *Senior Member, IEEE*

Abstract—BGP hijack attacks deflect traffic between endpoints through the attacker network, leading to man-in-the-middle attacks. Thus its detection is an important security challenge. In this paper, we introduce a novel approach for BGP hijacking detection that is based on the observation that during a hijack attack, the *functional* roles of ASNs along the route change. To identify a functional change, we build on previous work that embeds ASNs to vectors based on BGP routing announcements and embed each IP address prefix (AP) to a vector representing its latent characteristics, we call it *AP2Vec*. Then, we compare the embedding of a new route with the AP embedding that is based on the old routes to identify large differences. We compare our unsupervised approach to several other new and previous approaches and show that it strikes the best balance between a high detection rate of hijack events and a low number of flagged events. In particular, for a two-hour route collection with 10-90,000 route changes, our algorithm typically flags 1-11 suspected events (0.01-0.05% FP). Our algorithm also detected most of the previously published hijack events.

Index Terms—Internet security, BGP, IP hijack detection, deep learning, AP embedding.

I. INTRODUCTION

THE INTERNET consists of thousands of Autonomous Systems (ASes), each AS operated by an administrative domain such as an Internet Service Provider (ISP), a business enterprise, or a University. Each autonomous system is assigned a globally unique number, also called an Autonomous System Number (ASN), and advertises one or more IP address prefixes (APs). Interdomain Routing between ASes is determined by the Border Gateway Protocol (BGP). BGP is a path-vector routing protocol that uses routing update messages to propagate routing changes. Each routing update lists the entire AS path to reach an IP address prefix (AP) and carries one or more BGP attributes. BGP allows each AS to choose its own policy on accepting routes (according to its import policy), selecting the best routes, and announcing routes (according to its export policy). An AS routing policy is

usually determined by the commercial contractual relationship with other administrative domains.

Like many other basic Internet services, BGP was not designed to be secured, and as a result, attacks on BGP were designed to divert traffic to the attacker network. The attacker can then send the traffic to its original destination - forming a man-in-the-middle attack, or it can use the hijacked IP space for various nefarious activities such as sending spam or setting impersonation sites. In recent years, there have been many reports of BGP hijack attacks [1], [2], e.g., more than 40% of the network operators reported that their organization had been a victim of a hijack attack in the past [1]. Attacks are attributed both to governments and criminal organizations.

To hijack an AP, a BGP speaker can announce this AP as its own. If the owner with a larger AP announces this address space, then due to the longest prefix matching (LPM) rule, all the traffic to the hijacked AP will be diverted to the attacker network. If the owner announces the same AP (most likely a /24), only traffic from a portion of the origins will be diverted. This type of hijack attack can be detected quite easily by tracking changes in the origin AS of APs [3].

To avoid easy detection, one can hijack traffic by pretending to be on an attractive path towards the destination. This can be done by pretending to be a direct upstream provider of the hijacked AS or by announcing a route to peers (through exchange points) using the preference for peers over providers. This type of hijack attack is harder to detect, and it is the main motivation for our work.

It is important to note that routes may be deflected in unreasonable ways, also due to human error, not necessarily due to malicious acts. Even such benign deflections expose traffic to MITM attacks, e.g., by traversing networks, which are involved in espionage and may lurk for interesting data. It is almost impossible to know the cause of a deflection without knowing the intent behind people's actions. Thus, we follow previous work [4], and throughout this paper, we will also use the term BGP hijacking in the context of misconfiguration.

To tackle IP hijack attacks, two types of solutions have been proposed: reactive mechanisms that improve or amend the current routing protocols in order to make hijack attacks harder, or detection solutions that alert operators when a hijack occurs. The most notable reactive mechanisms to deal with the attack are RPKI [5] and BGPsec [6]. However, most operators have not deployed them and are reluctant to deploy them due to technical and financial costs [7], [8].

IP Hijack detection is a hard task, and previous work fails to provide solutions that are practical on a global scale. A

Manuscript received 6 September 2021; revised 15 January 2022; accepted 25 March 2022. Date of publication 11 April 2022; date of current version 12 October 2022. This research was funded in part by the Israeli PMO cyber grant program and by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University. The associate editor coordinating the review of this article and approving it for publication was C. Fung. (*Corresponding author: Tal Shapira.*)

Tal Shapira was with the School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel. He is now with the School of Computer Science, Reichman University, Herzliya 1096, Israel (e-mail: talshapira@gmail.com).

Yuval Shavitt is with the School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: shavitt@eng.tau.ac.il).

Digital Object Identifier 10.1109/TNSM.2022.3166450

number of papers [9], [10] suggested to alert when new AS links appear in a route, namely links that were not seen in the past. Gao [10] examined several heuristics and claimed as a best-case to have 0.02% false alarm per day for 2006 Internet, which is equivalent to over 6000 false alarms when analyzing a single RouteViews OIX file (with over 30M routes). The Artemis system [9] claims that for 28% of the ASes, there will be more than two false alarms per 3 days, namely in the current Internet, we will have about 15,000 false alarms per day. ARTEMIS also requires access to local monitors and other local ground truth, which may not be trivial. In general, ARTEMIS is designed to be used by a single operator locally, while we are looking at a global detection system. Other detection solutions rely on an increase in RTT [11] (which produces too many events); or identifying large bursts of BGP announcements [12], which can detect only large scale events.

We are only aware of one direction that is using deep learning for this IP hijack detection, termed BGP2Vec [13], and our work extends this approach. The main problem with BGP2Vec is that it requires a labeled dataset for training a deep learning system to detect a hijack attack. Obtaining such labeling is challenging.

In this paper, we introduce a novel approach for detecting BGP hijack attacks on a global scale, which is based on the observation that during a hijack attack, the ASes along the route have different functional roles than the usual route, which is true regardless of the hijack type. We term a route with functional changes of one or more of its ASNs, as a structurally-changed route. We also observe that when non-malicious route changes occur, the functionality along the route does not change. The functional role of an AS is not well defined. While in some cases it is easy to define the function that a certain AS plays, e.g., a tier-1 provider or a cloud provider, in general, an AS can have a more complex description of its function in the routing system, e.g., a tier-2 provider in Europe. To overcome the difficulty of defining the important functional classes and classifying the ASes into these classes, we use deep-learning-based embeddings that automatically perform this task.

To capture an ASN functionality, Shapira and Shavitt [13] build on the excellent results from Natural Language Processing (NLP) analysis where tools such as *Word2Vec* [14] are successfully used to map words in a language to their latent functional roles. Thus, they treated the BGP paths as sentences in NLP and introduced an extension of word vectors for creating a dense vector representation of ASNs (*BGP2Vec*). Using the *BGP2Vec* embedding, they trained a recurrent neural network using a labeled dataset to classify BGP routes as standard or hijacked routes. Our goal is to remove the dependence on the labeled dataset.

We suggest two methods to use the ASN embedding in order to identify functional changes along the route. First, to partition the ASNs into several functional classes (based on the embedding) and to examine changes in the functional roles of the ASNs along the path between two time periods. In the second method, we extend the concept of *BGP2Vec* and introduce *AP2Vec*, which embeds each AP to a dense vector. Similar to Paragraph Vectors [15] in the field of NLP, we use

routes as sentences and train each AP Vector to predict ASNs in their corresponding routes.

Unlike *BGP2Vec*, we do not use any labeled datasets and present an unsupervised system for BGP hijacking detection. In the first stage, we compare routes from two consecutive time samples based on the obtained ASN embedding and the AP embedding. We highlight APs whose route functional roles were changed (suspected routes). We compare several methods for this stage and find that the similarity between the embedding of a changed path with the embedding of its origin AP achieves the best balance between detection rate and false alarm rate. In the second stage, we identify hijacking events from the collection of suspected routes. We look for multiple suspected routes for the same AP with the same suspected hijacking ASN. We also aggregate multiple APs that are hijacked by the same ASN into a single event. Thus, a human operator is presented with a small number of suspected hijack events.

Our approach achieves excellent results. We test our algorithm on 13 past documented hijack events between 2008-2019 and detect correctly almost all the events. When examining random times, we show that our system flags only a few events, and we report two new recent events that the system detected. Finally, as far as we know, we are the first to present an unsupervised deep learning approach to detect IP hijack attacks based only on BGP announcements.

The rest of the paper continues as follows. After describing related work in Section II, we describe the dataset in Section III. In Section IV we describe the problem and give the motivation for our approach, and in Section V we introduce some preliminaries on Neural Network Embedding. In Section VI we describe our method; first, we describe our method for AP embedding, i.e., *AP2Vec*, then based on the obtained embedding, we describe our unsupervised method for BGP hijacking detection. In Section VII we explore the latent characteristics of the AP embedding vectors, and in Section VIII we investigate several past hijack events, analyze the performance of our method based on recent arbitrary time slots, and present quantitative results with comparison to previous methods. Finally, in the last two sections, we present technical considerations and conclude the paper.

To the best of our understanding, this work does not raise any ethical issues.

II. RELATED WORK

Approaches for defending against BGP hijacking can be sorted into two categories: prevention and detection. BGP prefix hijacking prevention solutions, also called proactive solutions, are based on cryptographic authentications such as RPKI and BGPsec [5], [6], [16], [17]. There are many different approaches for the detection of BGP hijacking. We divide these approaches into three main categories, based on the type of information they use: 1) Control-plane approaches [9], [18], [19] - also called passive solutions. These methods analyzed BGP routing information from a distributed set of BGP monitors and route collectors to detect anomalous behavior, 2) Data-plane approaches [2], [11], [20], [21] - only relies

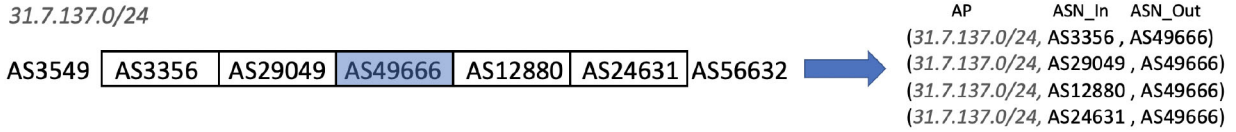


Fig. 1. An example for generating input/output ASNs and AP for the training process of *AP2Vec*.

on real-time data plane information that is obtained from multiple sensors that deploy active probing (pings/traceroutes). Some of these methods are based on analyzing IP TTL (Time to Live) or an increased RTT (round-trip delay time), and others are based on analyzing AS links changes, and 3) Hybrid approaches [22], [23], [24] - these approaches use both control-plane and data-plane information and sometimes also use external databases to perform joint analysis.

Several recent works [4], [25], [26], [27], [28] examined machine learning techniques with manually generated features to identify malicious origin ASes; mainly leveraging historical BGP data from RouteViews [29] and RIPE. Testart *et al.* [28] focused on identifying dominant characteristics of serial hijackers over a long-term period of 5 years (such as intermittent AS presence, short prefix origination duration, etc.). They generated dominant features, used a tree-based machine learning classifier model for identifying ASes with BGP origination patterns similar to serial hijackers, and identified about 900 ASes with suspected malicious behavior.

Cho *et al.* [4] classified hijack events in order to understand the nature of reported events. They introduced four categories of BGP hijack attacks: typos, prepending mistakes, origin changes, and forged AS paths. Unlike previous works, their work aimed to classify detected hijack events and not detect new hijack events. They generated five features, including AS hegemony [30] (a metric that quantifies the likelihood of an AS to lie on paths toward particular destination IP prefixes), and used a Random Forest classifier, which achieved an accuracy of 95.71% over their dataset.

In recent work, Moriano *et al.* [12] proposed an algorithm that leverages the burstiness of disruptive BGP updates to provide early detection of large-scale malicious incidents using local collector data. They validated the effectiveness of their method by analyzing four cases of large-scale routing anomalies. However, their method is less effective in detecting small-scale events.

III. THE DATASET

As mentioned in Section I, our approach is based on analyzing BGP announcements. For this end, we use the RouteViews' BGP paths dataset (RV) [29], which contains BGP paths collected from about 40 vantage points.

A challenge in studying BGP hijacking is the limited availability of ground truth hijack events. Therefore, in order to validate our results, we collected 13 BGP events from several blogs including Dyn [31], RIPE [32], BGPMon [33], APNIC [34], ORACLE [35], MANRS [36], and BGProtect [37] as sources for hijack/leak events that were manually checked by experts and reported as suspicious. The events are from February 2008 to June 2019 (see Table III,

and they are all long enough to be captured in the RV oix files. Among the events, 7 events were also analyzed by Cho *et al.* [4], and 3 were analyzed by Moriano *et al.* [12]. Furthermore, we also test our approach using several arbitrary samples from October 2020 (see Table IV) in order to test the False Alarm behavior.

For each event, we retrieve both the RV's file corresponding with the start time of the event and one RV file before. For example, for an event that started at 2018-04-24 11:05, we use the files of 2018-04-24 12:00 and 2018-04-24 10:00; each contains BGP announcements from two hours.

IV. MOTIVATION AND PROBLEM STATEMENT

We consider the problem of BGP hijacking detection by detecting hijacked APs and a hijacker ASN. More formally, let \mathcal{P}_t be a collection of AS-level paths (e.g., BGP announcements) and \mathcal{T}_t their corresponding origin APs, which are collected at time t , such the each path $p \in \mathcal{P}_t$ consists of a series of ASNs and has a corresponding origin $AP \in \mathcal{T}_t$, as demonstrated in Figure 1. Our objective is to detect a collection of hijacked paths p_i, \dots, p_{1+l} , their origin AP and the hijacker ASN, which is involved in all of the hijacked paths.

Shapira and Shavitt [13] showed that they could use supervised learning to differentiate between a BGP hijack event and a benign route change since in a benign routing, changed ASNs are replaced with functionally similar ones. However, in a hijack attack, the sequence of ASN functions along the path changes. In many cases, this may result in the violation of valley-free routing [10], so this structural change detection can be viewed as a generalization of detecting valley-free violation.

We extend the method that was introduced in [13] and propose an unsupervised method, which learns simultaneously representations of ASNs and APs that capture the AS-level graph structure based entirely on BGP announcements. Consequently, we utilize the significant information about the dependence of ASNs to capture the functional role of each ASN, as well as the similarity between a path and an origin AP.

Perozzi *et al.* [38], motivated their usage of NLP models for social graph representation learning by their similarity, as both word distribution in a language and social graphs have power-law degree distribution. For the Internet AS-level graph, Faloutsos *et al.* [39] were the first to show that the inter-AS topology exhibits a power-law (scale-free) node degree distribution, which was later shown to be true for extended connectivity data by Shavitt and Shir [40].

Natural language models are trained using large amounts of unstructured text data, which in our case are equivalent to a collection of BGP announcements (an AS-level path is

equivalent to a sentence). The word representations, which are computed using neural networks, capture many linguistic and latent characteristic [41]. Paragraph vectors [15], which are inspired by the word vectors, act as a memory that remembers what is missing from the current context. In the same way, an AP acts as a memory that remembers what is missing from the current context of ASNs in BGP routes.

Our goal is to simultaneously learn a latent representation for ASNs, which is defined by the embedding function $\Phi: v \in \mathcal{V} \mapsto \mathbb{R}^{d_v}$, and a latent representation for APs, which is defined by the embedding function $\Psi: u \in \mathcal{U} \mapsto \mathbb{R}^{d_u}$, where d_v , d_u are small numbers of latent dimensions. These mappings Φ , Ψ represent the latent representation associated with each ASN and each AP, correspondingly. For a given path, we can generate a low-dimensional representation based on its ASNs and calculate the similarity to its origin AP. We claim that the distance between a representation of a hijacked route and the origin AP is much larger than the distance of a legitimate path.

V. PRELIMINARIES

This section introduces some preliminaries on Neural Network Embedding, which is the main ingredient of our approach. Applications of neural networks have expanded significantly in recent years [42]. One of them is embedding discrete values to continuous N-dimensional vectors. This technique is broadly used in the field of Natural Language Processing (NLP), also known as word embedding [14], [41] or Neural Language Modeling, which helps machine learning algorithms to achieve better performance by grouping similar words. Embedding is important for input to a neural network, as it is trained to work on vectors of real numbers. Moreover, the method produces vectors such that similar words have close vectors, where similarity is defined in terms of both syntax and semantic.

Word2Vec [14] is one of the most popular unsupervised data-driven techniques to learn word embedding. This technique uses a shallow neural network fed with large amounts of unstructured text data. Word2Vec has two important benefits; first, its representations exhibit linear structure that makes precise analogical reasoning possible. For example: $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman}$ resembles the closest vector to the word *Queen* [41]. The second benefit is that in contrast to one-hot encoding, which maps each word to a vector with a size equal to the number of unique words in the corpus, and therefore makes training any machine learning model on this representation infeasible, Word2Vec maps each word to a vector of size d , with d significantly smaller relative to language size.

There are two kinds of Word2Vec models, which have opposite training objectives: Skip Gram and Continuous Bag Of Words (CBOW). The skip-gram model takes a word as input and seeks to predict the surrounding words in a sentence or a document. In contrast, the CBOW model takes each word's context as the input and tries to predict the word corresponding to the context.

We will focus on describing the CBOW model. In this model, we analyze each sentence in the corpus and train the model to predict each word in the sentence using the words

within a certain range before and after the current word as inputs. As a result, the model learns to characterize a word by its context, i.e., neighboring words. The model itself consists of a neural network with a single hidden layer. The input layer is of size $|\mathcal{V}|$, equal to the number of unique words in the corpus (or vocabulary), such that a unique one-hot encoding vector represents each word. The hidden layer is a fully-connected layer of size equals to the embedding size d . Within a *fully-connected layer*, neurons between two adjacent layers are fully pairwise connected, while neurons within the same layer share no connections. Each artificial neuron performs a summation of the dot product of its inputs and its weights and adds a bias term. The hidden layer weight matrix is of size $|\mathcal{V}| \times d$, such that each word in the corpus corresponds to a d -features vector. These are the word vectors that are learned by the model. Therefore, the output of the hidden layer is the embedding for the input word. The output layer is the *softmax layer*, which is a generalization of the logistic function that normalized a vector of arbitrary real values to a probability distribution vector of real values in the range $[0, 1]$ that add up to 1. In *Word2Vec* the softmax layer is of size $|\mathcal{V}|$ (the number of unique words in the corpus) for each desired output.

Inspired by the words embedding models, **Doc2Vec** [15] aims to learn *paragraph vectors*. Here there are also two kinds of models, which have opposite training objectives: Distributed Memory (DM) and Distributed Bag of Words (DBOW). We will focus on describing the DM model, which is the one we use in *AP2Vec*. In this model, every paragraph (i.e., sentence or a collection of sentences) is mapped to a unique vector, and simultaneously every word is also mapped to a unique vector, using two separated hidden layers (one for words and one for the paragraph). The paragraph vector and word vectors are averaged (or concatenated) to predict the word which is surrounded by the context words, as shown in Figure 1 for AS-path.

The training stage is done by feeding the neural network with words and a paragraph token; the inputs are one-hot vectors representing the context words and one-hot vector representing the paragraph, and the training output, which is also a one-hot vector representing the output word. Then applying gradient descent learning [43] (also known as back-propagation) to adjust the weights of the network in order to maximize the log probability of the output word given the context words and the paragraph.

One difference from the word embedding model happens in the inference stage. In *Doc2Vec* it is possible to generate vectors for new paragraphs (i.e., never seen in the training stage) by adding more columns to the hidden layer of the paragraph vectors ($\Psi_{(|\mathcal{U}|+1) \times d}$) and applying gradient descent while holding the rest of the neural network's weights fixed ($\Phi_{|\mathcal{V}| \times d}$ and $\Phi'_{d \times |\mathcal{V}|}$).

VI. METHOD

We aim to identify BGP hijack attacks where the hijacker injects its ASN into the route to deflect the traffic toward its network. To this end, we perform the following stages (see Figure 3):

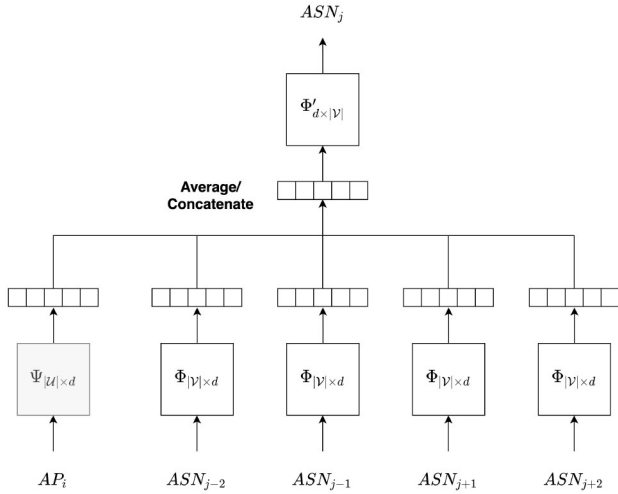


Fig. 2. Our AP2Vec architecture, where $|\mathcal{V}|$ is the number of ASNs, $|\mathcal{U}|$ is the number of APs, $d = 32$, and $w = 2$.

- 1) We map both ASNs and APs to a latent space (we choose 32 dimensions following *BGP2Vec* [13]) that captures functional and geographical similarities.
- 2) We identify routes that have a significant intrinsic structural change from previous routes and then flag the APs with many such changes that are due to the same ASN injection.

In this section, we will introduce in detail both stages.

A. AP2Vec

As mentioned in Section I, in the first stage, we produce a d -dimensional continuous vector representation for each ASN and each AP. As in the training process of document embedding in NLP, i.e., Doc2Vec [15], we train our network over a large corpus of APs \mathcal{T} , such that each AP has several BGP paths from \mathcal{P} (described in Section III), one for each vantage point. The APs and their paths are equivalent to documents and their sentences in NLP tasks. We apply a similar *Distributed Memory Model of Paragraph Vectors* (PV-DM) as introduced in [15], such that we predict each ASN in a certain AS-path based on the corresponding AP and ASNs within a certain range before and after the target ASN (as shown in Fig. 2). As a result, the model learns in parallel both to characterize an AP by the ASNs used to reach it and an ASN by its context (a combination of neighboring ASNs and the origin AP), i.e., the model embeds both ASN-vectors and AP-vectors simultaneously. This is one of the main reasons for choosing the PV-DM model for this task.

As presented in Algorithm 1 We begin by building vocabularies of all the ASNs (line 2) and the APs (line 3) in the dataset and randomly initializing the embeddings for all ASNs ($\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$) (line 4) and APs ($\Psi \in \mathbb{R}^{|\mathcal{U}| \times d}$) (line 5) in the vocabularies. Then for each path and its corresponding AP, we apply the PV-DM model (Algorithm 2) as follows. Given an AS-level path $\{v_1, v_2, v_3, \dots, v_N\}$ and a context window of size w , the PV-DM model attempts to predict each origin AS v_j by its context C , which is defined by its surrounding ASes

Algorithm 1: AP2Vec($\mathcal{P}, \mathcal{T}, w, d, \alpha$)

inputs: \mathcal{P} : AS paths list
 \mathcal{T} : corresponding origin APs
 w : window size
 d : embedding size
 α : learning rate
output: matrix of ASN representations $\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$
matrix of AP representations $\Psi \in \mathbb{R}^{|\mathcal{U}| \times d}$

```

1 begin
2   Initialization: Generate vocabulary  $\mathcal{V}$  from  $\mathcal{P}$ ,
3   Generate vocabulary  $\mathcal{U}$  from  $\mathcal{T}$ ,
4   Sample  $\Phi$  from  $\mathbb{R}^{|\mathcal{V}| \times d}$ ,
5   Sample  $\Psi$  from  $\mathbb{R}^{|\mathcal{U}| \times d}$ 
6   for each  $path \in \mathcal{P}$  &  $u_i \in \mathcal{T}$  do
7     | PVDM( $\Phi, \Psi, path, u_i, w, \alpha$ )
8   endfor
9   return  $\Phi, \Psi$ 
10 end
```

Algorithm 2: PVDM($\Phi, \Psi, path, u, w, \alpha$)

```

1 for each  $v_j \in path$  do
2   |  $J(\Phi, \Psi) =$ 
3   |  $-\log \Pr(v_j | \Phi(v_{j-w}), \dots, \Phi(v_{j+w}), \Psi(u))$ 
4   |  $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$ 
5   |  $\Psi = \Psi - \alpha * \frac{\partial J}{\partial \Psi}$ 
6 endfor
```

and corresponding AP u_i :

$$C = \{v_{j-w}, \dots, v_{j-1}, v_{j+1}, \dots, v_{j+w}, u_i\}. \quad (1)$$

More formally the objective of the PV-DM model is to learn the latent representations $\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\Psi \in \mathbb{R}^{|\mathcal{U}| \times d}$, which maximize the average log co-occurrence probability:

$$\frac{1}{N} \log \Pr(v_j | C; \Phi, \Psi). \quad (2)$$

As presented in [14], the basic neural-network based Skip-Gram formulation defines the conditional probability $\Pr(v_{j+k} | v_j)$ using the *softmax* function:

$$\Pr(v_{j+k} | v_j) = \frac{\exp(\Phi'_{v_{j+k}} \top \Phi_{v_j})}{\sum_{v \in \mathcal{V}} \exp(\Phi'_v \top \Phi_{v_j})} \quad (3)$$

where Φ_v and Φ'_v are the input and output vectors of AS v . In our case of the PV-DM model there is a hidden layer that averages the context vectors:

$$h = \frac{1}{|C|} \sum_{v \in C} v. \quad (4)$$

Then the conditional probability $\Pr(v_j | C)$ is defined by:

$$\Pr(v_j | C; \Phi, \Psi) = \frac{\exp(\Phi'_{v_j} \top h)}{\sum_{v \in \mathcal{V}} \exp(\Phi'_v \top h)} \quad (5)$$

However, this formulation is impractical because the computation cost of the conditional probability is proportional to $|\mathcal{V}|$,

which in our case is very large ($10^5 - 10^6$ terms). Therefore we use the negative-sampling approach, as introduced in [14], as a more efficient way of deriving ASN embeddings, by replacing the log-likelihood term in the PV-DM objective with the term:

$$\log \sigma(\Phi'_{v_j}{}^\top h) + \sum_{i=1}^k \mathbb{E}_{v_i \sim P_n(v)} [\log \sigma(-\Phi'_{v_i}{}^\top h)], \quad (6)$$

where σ is the *Sigmoid* function, which is defined by $\sigma(x) = 1/(1 + \exp(-x))$. Negative sampling selects k ASNs that are not in the context at random noise distribution $P_n(w)$ instead of considering all ASNs in the graph. In our experiments we used the same noise distribution as described in [14].

Our neural network has two types of inputs: $2w$ inputs for the ASNs in the context window and one for the AP. Each input is embedded into a d -feature vector with an embedding matrix $\Phi_{|\mathcal{V}| \times d}$ for ASNs and $\Psi_{|\mathcal{U}| \times d}$ for APs; these are the ASN-vectors and AP-vectors that are learned by the model. The embedding layers are connected by an average operation (the hidden layer) of size d . The last layer is the output layer of size $|\mathcal{V}|$ for the desired output ASN, with a weight matrix $\Phi'_{d \times |\mathcal{V}|}$.

The hyper-parameters we choose, are based on optimization for *BGP2Vec* [13]: the window w is set to 2 (see Figure 1), and the embedding size is set to 32. In order to improve the representation, we use 5 negative samples [14] for each target ASN. We build and run our network using the *Gensim* [44] library. The training procedure is done by feeding the network with pairs of context C and target ASN v_j ; the inputs are one-hot vectors representing the input ASNs and AP and the training output, which is also a one-hot vector representing the output ASN. Then applying gradient descent learning [43] (also known as back-propagation) to adjust the weights of the network, in order to maximize the log probability of any target ASN given the context ASNs and corresponding AP based on Equations (2), (6). We repeat this process for 3 epochs.

B. BGP Hijacking Detection

Based on the *AP2Vec* model, we can now describe the main method for BGP hijacking detection. Given two consecutive RV records, we apply the *AP2Vec* model on the first record and obtain both ASN-vectors and AP-vectors. For each AP $u' \in \mathcal{T}_t$ we define its *parent AP* $u \in \mathcal{T}_{t-1}$ as the minimal super-prefix of the current AP. This is because one of the most common and problematic hijack attacks is the ‘more-specific origin change,’ in which a hijacker announces a sub-prefix of the prefix announced by the legitimate AS, therefore in this case, the AP is not part of the preceding record \mathcal{T}_{t-1} . Be aware that in the case that the AP is part of the preceding record, it is also equal to the *parent AP*. It is worth mentioning that our method can not be applied in a case where there is no *parent AP*, which is the case of unallocated or unused APs.

At the first phase of our unsupervised method, we compare each path $p' \in \mathcal{P}_t$ in the second record, with the path $p \in \mathcal{P}_{t-1}$, which shares the same vantage point (ASN) and *parent AP*, from the first record. If the paths differ ($p' \neq p$), we

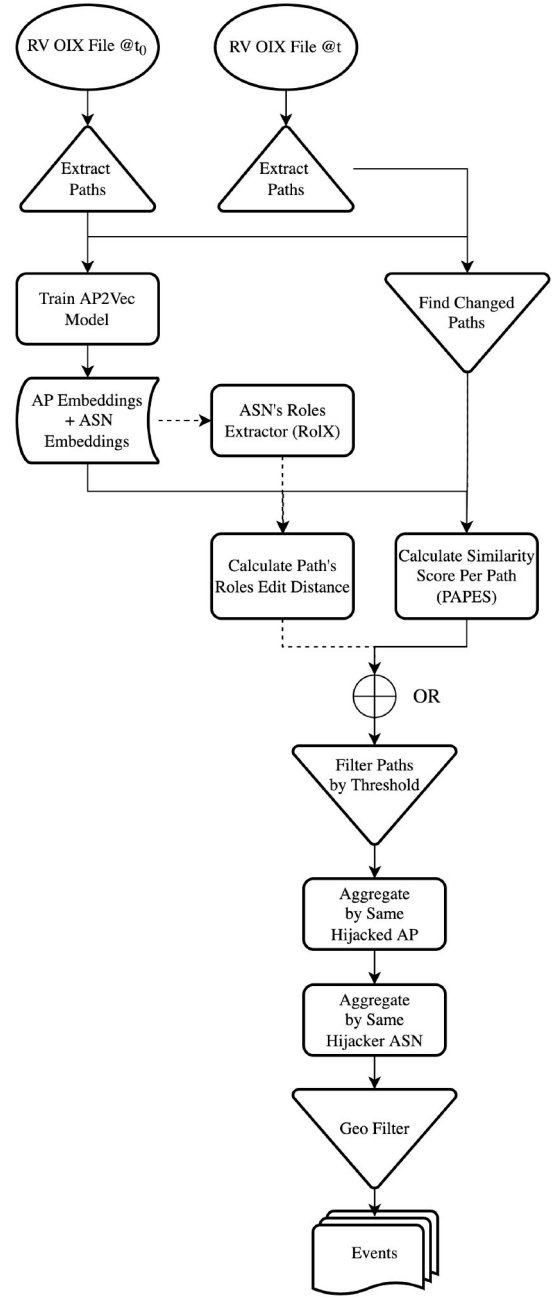


Fig. 3. A block diagram of our BGP hijacking detection system. The dashed arrows depict an alternative (with slightly higher FA rate) method.

propose two methods to detect a structural change in the path (see Figure 3):

- 1) *Similarity between a path and an AP (PAPES)*: as mentioned in Section V, one of the main advantages of the *Doc2Vec* model is its ability to generate vectors for new paragraphs, and in our case a new AP or even specific paths. This made by adding more columns to the hidden layer of the AP vectors ($\Psi_{(|\mathcal{U}|+1) \times d}$) and applying gradient descent while holding the rest of the neural network's weights fixed ($\Phi_{|\mathcal{V}| \times d}$ and $\Phi'_{d \times |\mathcal{V}|}$). Then, for a given path, we generate its embedding vector (we choose to run 50 epochs of inference) and calculate the similarity between the generated vector and the

$AP2Vec$ vector of its parent AP Ψ_u . We define similarity between two $AP2Vec$ vectors $\Psi_{u_i}, \Psi_{u_j} \in \mathbb{R}^d$ of the corresponding APs $u_i, u_j \in \mathcal{U}$ as the *cosine similarity*:

$$\text{sim}(\Psi_{u_i}, \Psi_{u_j}) = \frac{\Psi_{u_i}^T \Psi_{u_j}}{\|\Psi_{u_i}\|_2 \cdot \|\Psi_{u_j}\|_2}, \quad (7)$$

and choose a maximal threshold th_{cs} which determines if the structural change of the path is significant, i.e., if $\text{sim}(\Psi_u, \Psi_{p'}) < th_{cs}$.

- 2) *Path's roles edit-distance* (Roles-ED): we replace each ASN in a path p' with its corresponding role and remove consecutive duplicates. We apply the same for the preceding path p , compute the edit-distance between the two role-paths p'_r, p_r using the Levenshtein distance [45] (\mathcal{LD}), and choose a minimal threshold th_{ld} which determines if the structural change of the path is significant. In order to generate the functional role of each ASN, we apply the unsupervised method for role extraction *RoIX* described in [46] with a slight modification. Instead of using the *ReFeX* algorithm [47] for recursive structural feature extraction, we use the ASN-vectors as a set of d structural features; one feature for each dimension of the embedding. In Section VIII we compare between the two methods and show that the usage of ASN-vectors as features outperforms the regular features. Using the Minimum Description Length criterion [48], *RoIX* automatically determines the number of roles that results in best compression. In all of our experiments, over the years 2008-2020, we obtain the same number of roles (in both methods) – 8.

As we present in Section VIII-A, we choose both thresholds based on the mean and standard deviations of the distribution of each metric.

In the second phase, we generate a list of candidate hijacked APs. These are APs that all their corresponding changed paths (denoted by $C_{u'}$) are due to a structural change. We denote by $L_{u'}$ the set of routes with a structural change from the previous phase and keep only the APs where $L_{u'} = C_{u'}$. Namely, we require that all routing changes for this AP will have a structural change. This requirement may be relaxed to $|L_{u'}| \geq \alpha |C_{u'}|$, for some α close to 1. Then, for each AP in the remaining list, we look for a newly seen ASN the appears in all the paths in $L_{u'}$ (lines 21-22 in Algorithm 3). If such an ASN does not exist, we remove the AP from the candidate list. This phase's outcome is a list of ASN hijackers L_{ASN} and their corresponding hijacked APs L_{AP} . The rationale for this phase is that a successful hijack attack should be observed from several vantage points, and no other routing change should occur concurrently.

Identifying the attacker ASN is not a trivial task, since in many cases, its BGP neighbor (upstream in many cases) also appears in almost all paths. Thus, at the last stage, look for ASNs that appear as suspected attackers for several APs. We unite all these APs to a single hijack event.

Algorithm 3: $\text{HijackDetection}(\mathcal{P}_{t-1}, \mathcal{P}_t, \mathcal{T}_{t-1}, \mathcal{T}_t, w, d, \alpha, th_{cs}, th_{ld}, th_{cu}, th_{cv})$

inputs: $\mathcal{P}_{t-1}, \mathcal{P}_t$: AS path lists at times $t-1, t$

$\mathcal{T}_{t-1}, \mathcal{T}_t$: corresponding origin APs

w : window size

d : embedding size

α : learning rate

output: L_{ASN} : list of suspicious ASNs

L_{AP} : the corresponding potential victim APs

```

1 begin
2   Initialization: Generate vocabulary  $\mathcal{V}$  from  $\mathcal{P}_{t-1}$ ,
3   Generate vocabulary  $\mathcal{U}$  from  $\mathcal{T}_{t-1}$ ,
4    $\Phi, \Psi \leftarrow AP2Vec(\mathcal{P}_{t-1}, \mathcal{T}_{t-1}, w, d, \alpha)$ 
5   Generate parent APs  $\mathcal{S}: u' \in \mathcal{T}_t \mapsto u \in \mathcal{U}$ 
6   for each  $p' \in \mathcal{P}_t$  &  $u' \in \mathcal{T}_t$  do
7     Get the corresponding  $p \in \mathcal{P}_{t-1}$  &  $u \in \mathcal{U}$ 
8     if  $p' \neq p$  then
9        $C_{u'} \leftarrow p$ 
10      Generate  $\Psi_{p'}$  for  $p'$ 
11      if  $\text{sim}(\Psi_u, \Psi_{p'}) < th_{cs}$  then
12         $L_{u'} \leftarrow p$ 
13      end
14    end
15  endfor
16  for each  $u' \in \mathcal{T}_t$  do
17    if  $L_{u'} == C_{u'}$  then
18      for each  $v \in \text{set}(L_{u'})$  do
19        if  $L_{u',v} == L_{u'}$  then
20           $L_{AP} \leftarrow u'$ 
21           $L_{ASN} \leftarrow v$ 
22        end
23      endfor
24    end
25  endfor
26  return  $L_{AP}, L_{ASN}$ 
27 end

```

VII. EXPLORATION OF AP EMBEDDINGS

We base $AP2Vec$ on $BGP2Vec$ [13], [49] where ASN embedding using BGP announcements was thoroughly explored. They showed that the ASN representations exhibit linear structure, and specifically, an ASN could be characterized by its closest ASNs. We claim here that these results also apply to the AP representations and show in this section qualitative examples (Evaluation of the embedding for hijack detection will be done in Section VIII). We found high cosine similarity between APs, which share the same function: universities, organizations, IXPs, content providers, geographical locations, and ASNs. We give two examples to illustrate the latent characteristics of the obtained AP vectors.

We define similarity between two $AP2Vec$ vectors according to Equation (7) and find the nearest neighbor of $AP2Vec$ vector $\Psi_{u_i} \in \mathbb{R}^d$ using:

$$\text{NearestNeighbor}(u_i) = \arg \max_{u_j \in \mathcal{U}} \text{sim}(\Psi_{u_i}, \Psi_{u_j}). \quad (8)$$

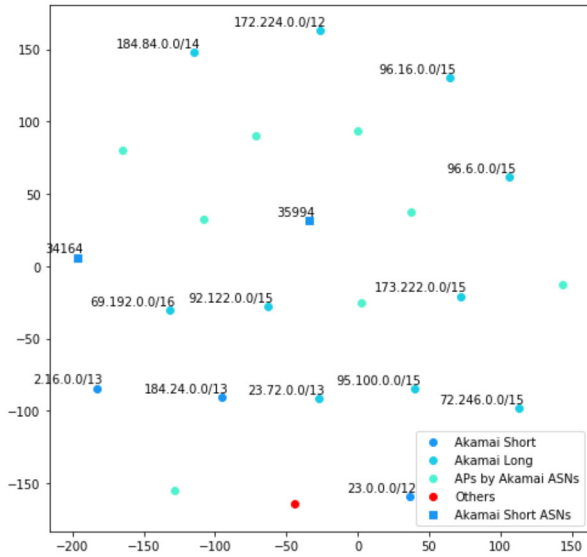


Fig. 4. Two-dimensional UMAP projection of 32-dimensional vectors of Akamai APs, ASNs and their nearest neighbours.

The same is also applied for ASN vectors. We calculate the *average cosine similarity correlation* of a specific group of APs or ASNs by taking the average of all the cosine similarities of the pairs in this group.

We use [50] to collect the list of Akamai and DigitalOcean APs and ASNs. For Akamai, we select three APs; ‘184.24.0.0/13’, ‘2.16.0.0/13’, and ‘23.0.0.0/12’ to a short-list termed ‘Akamai Short,’ and term the rest of the APs as ‘Akamai Long’ list. We then find the 10 nearest neighbors for each of the three APs in ‘Akamai Short.’ Figure 4 presents a 2-dimensional UMAP [51] projection of the obtained vectors. As presented, among the nearest neighbors, we can find 10 additional APs which belong to ‘Akamai Long,’ and there is only one AP among the nearest neighbors which do not belong to Akamai. Furthermore, the two ASNs which own the three Akamai APs are located close to Akamai’s APs. The average cosine similarity correlation of Akamai APs is 0.595, whereas the average cosine similarity correlation of a random sample of 1000 APs is 0.317, which means the representations capture the closeness of Akamai APs.

For DigitalOcean, we find the 10 nearest neighbors for an arbitrary set of 28 of its APs. Figure 5 depicts a 2-dimensional UMAP [51] projection of the obtained vectors. As presented, all the nearest neighbors are owned by DigitalOcean ASNs. This result is consistent with the high cosine similarity correlation obtained for DigitalOcean APs, which is equal to 0.96.

Figure 6 depicts a UMAP projection of both Akamai and DigitalOcean APs. There is a clear separation between the APs of these ASNs, which can be explained by the low average similarity between them, only 0.18; much lower than 0.312, which is the average similarity between random ASNs. The low similarity between these two ASNs is surprising since both are global CDNs.

Figure 7 depict APs from universities around the world: MIT from the US; U. of Cambridge and U. of Oxford from

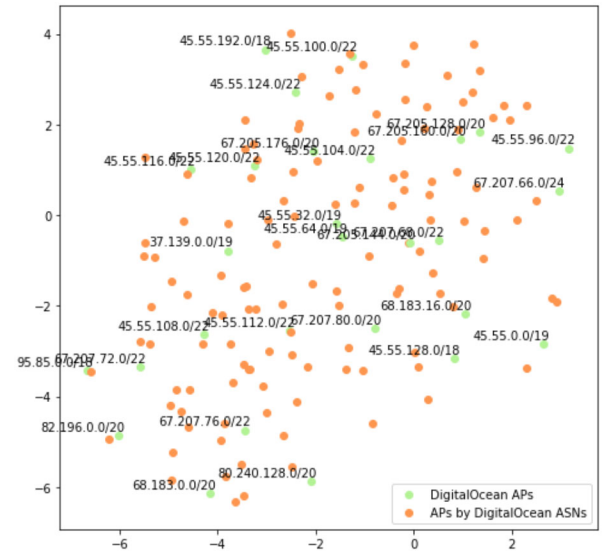


TABLE I

SUMMARY OF METRIC STATISTICS BASED ON 4 CHOSEN TIMESTAMPS. METRICS ARE CALCULATED ONLY FOR APs WITH CHANGED PATHS, AND ONLY FOR THE PATHS THAT WERE CHANGED. WE PRESENT THE MEAN AND STANDARD-DEVIATION ($\mu \pm \sigma$) ACROSS ALL APs

Record Time	#APs	#ASNs	#APs w/ Changed Paths	PAPES	Roles-ED	F-Roles-ED	Local Hegemony Sim	Valley Diff
2020-10-18 16:00	883,337	70,334	10,208	0.46 ± 0.25	0.82 ± 0.77	0.57 ± 0.76	0.82 ± 0.17	-0.02 ± 0.24
2018-04-24 12:00	753,146	61,206	15,605	0.51 ± 0.26	0.56 ± 0.70	0.17 ± 0.46	0.87 ± 0.12	0.13 ± 0.24
2014-11-15 00:00	542,384	48,881	11,404	0.51 ± 0.25	0.96 ± 0.49	0.72 ± 0.68	0.90 ± 0.09	-0.02 ± 0.23
2008-02-24 20:00	246,271	27,478	16,043	0.25 ± 0.26	1.20 ± 0.63	0.93 ± 0.81	0.91 ± 0.09	0.00 ± 0.15

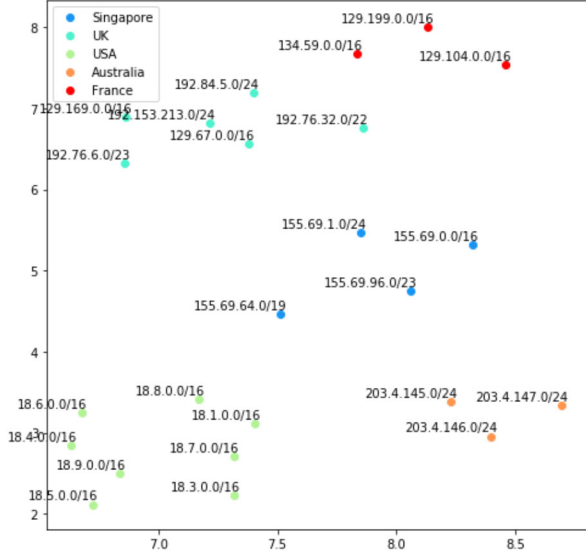


Fig. 7. Two-dimensional UMAP projection of 32-dimensional vectors of University APs.

by European ASNs, and role 7 has a significant portion of Asian ASNs.

It is also interesting to note that three roles (0, 5, and 7) have 3-4% of the ASNs, four have 9-15% of the ASNs, and role 1 has 42% of the ASNs. When we forced the system to divide the ASNs into 32 roles (the maximum possible), role 1 still holds 25% of the ASN. Obviously, there are many small ASes that are hard to distinguish and end up in this role.

VIII. EXPERIMENTS AND RESULTS

In this section, we report our experimental results for BGP hijacking detection. As mentioned in Section III, we evaluate our two method's ability to detect real events by testing it using a list of ground truth past events. To evaluate the false alarm rate, we test our methods using several arbitrary RV records from October-November 2020.

To show the advantage of using deep learning for this task, we compare our two suggested methods (see Section VI-B) with recently proposed path-based metrics, which are entirely based on BGP paths without any side information. First, as mentioned in Section VI-B, for role extraction, we compare the usage of ASN-vectors as features and the usage of the default recursive structural features (*ReFeX* [47]) as used in *RoIX*. Second, we propose two other methods to detect a structural change in a path, based on the global AS hegemony [30] and local AS hegemony that were presented in [4]. The global AS hegemony is computed based on paths to all APs globally

reported by the vantage points, while the local AS hegemony is computed based on paths towards only one origin AS. We calculate AS hegemony based entirely on RV records and do not use any external source. Therefore we compare a total of 5 metrics for the detection of structural changes:

- 1) *Path and AP Embedding Similarity* (PAPES)- as defined in Section VI-B.
- 2) *Path's roles edit-distance using the ASN-vectors as features* (Roles-ED) - as defined in Section VI-B.
- 3) *Path's roles edit-distance using ReFeX [47] features* (F-Roles-ED).
- 4) *AS local hegemony similarity* (Local Hegemony Sim) - as described in [4]. We select the top three ASes in terms of local hegemony scores from the preceding paths and current paths and calculate the cosine similarity between them. Note that, unlike the other metrics that are computed on a single path, this metric is calculated over the entire list of changed paths of the origin AP.
- 5) *AS global hegemony valley depths difference* (Valley Diff) - we extend the method described in [4] for calculating the valley depth for a path based on global hegemony and subtract the depth of the preceding path from the depth of the current path. Since Cho *et al.* [4] mentioned several limitations of using AS hegemony for BGP hijacking classification, we choose to use the difference between two consecutive valleys and not the absolute value to minimize these limitations.

A. Parameters Exploration

As mentioned in Section VI-A, we choose the hyper-parameters for the *AP2Vec* model based on the hyper-parameters optimization conducted in [13].

For the hijack detection stage, we explore our chosen metrics based on 4 experiments with the following timestamps: 2020-10-18 16:00, 2018-04-24 12:00, 2014-11-15 00:00, and 2008-02-24 20:00. Table I presents statistics for the five metrics. Notice that for each AP, we calculate the average metric only across its changed paths. We use these statistics to define the thresholds mentioned in Section VI-B.

Then, for each similarity metric (see Table I), we choose an upper threshold of $\mu - \sigma$, and for each distance metric, we choose a lower threshold of $\mu + \sigma$ (in the case of a perfect normal distribution, this leaves us with 15.86% of the results). Since the Edit-Distance is an integer, we round down the threshold (e.g., for Roles-ED, we round down 1.59 to 1, meaning that we consider a structural change in a path if the edit distance is at least 1). Additional analysis of the statistics of the metrics appears in the Appendix.

TABLE II

SUMMARY OF THE NUMBER OF FLAGGED CANDIDATES BASED ON DIFFERENCE METRICS FOR THE 4 CHOSEN TIMESTAMPS. HERE, WE DID NOT REMOVE EVENTS WITH ASNs WHICH SIGNIFICANTLY APPEARED IN THE PREVIOUS TIME-SLOT ANALYSIS

Record Time	PAPES			Roles-ED			F-Roles-ED			Local Hegemony Sim			Valley Diff		
	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs
2020-10-18 16:00	10	48	16	41	163	67	42	101	65	20	54	33	4	8	10
2018-04-24 12:00	16	73	21	60	223	88	39	134	55	40	174	60	6	50	10
2014-11-15 00:00	5	14	6	23	52	31	30	71	42	26	62	35	2	2	2
2008-02-24 20:00	2	10	5	46	121	70	31	93	43	40	74	61	6	10	7

TABLE III

SUMMARY OF EXPERIMENTS ON GROUND TRUTH EVENTS. METRICS REPRESENT THE MEAN VALUE ACROSS ALL AFFECTED APs. WE MARK WITH BOLD VALUES OF METRICS WHICH DETECT THE EVENT

#	Event Name	Event Time	Hijacker	APs w/ Changed Paths	PAPES	Roles ED	F-Roles ED	Valley Diff	Local Hegemony Sim
1	<i>Level3 Leak</i>	2017-11-06 17:47	AS3356	6,569	0.18	1.08	1.04	0.49	0.63
2	<i>Amazon/Route 53</i>	2018-04-24 11:05	AS10297	5	0.17	1.88	0.00	-0.14	0.94
3	<i>Bangladesh Event</i>	2018-09-26 07:45	AS17494	2,622	0.05	1.38	1.71	0.54	0.67
4	<i>YouTube Hijacking</i>	2008-02-24 18:47	AS17557	1	0.115	0.86	0.36	0.00	0.68
5	<i>Hijacks by Indosat</i>	2014-04-02 18:26	AS4761	1,534	0.07	1.48	1.13	0.00	0.68
6	<i>Telecom Malaysia leak</i>	2015-06-12 08:43	AS4788	212,368	-0.11	1.51	1.85	0.47	0.62
7	<i>Hijack in India</i>	2015-11-06 05:52	AS9498	1,086	0.18	1.05	1.05	0.45	0.79
8	<i>Trouble in Switzerland</i>	2019-06-06 09:43	AS21217	1,715	0.02	3.03	3.28	0.95	0.55
9	<i>Iran Leaks Censorship</i>	2017-01-05 11:30	AS12880	1	0.17	1.92	2.5	0.05	0.76
10	<i>Petersburg 1</i>	2015-01-07 09:00	AS44050	1	0.13	1.16	2.03	0.05	0.34
11	<i>Bitcanal Hijack 1</i>	2018-06-29 13:00	AS197426	1	0.22	1.11	1.81	-0.03	0.76
12	<i>Bitcanal Hijack 3+4</i>	2015-01-07 12:01	AS197426	4	0.09	2.45	1.18	0	0.93
13	<i>H3S Hijack</i>	2014-11-14 23:00	AS59790	2	0.41	1.22	1.94	-0.03	0.68

Based on the chosen thresholds for each metric (excluding local hegemony similarity) and for each AP, we count the number of structural-changed paths. For all the metrics we use, more than 80% of the APs with path changes have less than 5 structurally-changed paths. If only a small number of paths were changed, it means that the deflection/hijack did not succeed in propagating through the Internet. Thus, we consider only APs with a minimum of 5 changed routes. We validated that all the ground-truth events are above this threshold.

Since we are interested in hijack attacks, there should be a cause for the route change. Thus, we require that in all the changed paths, there will be a role change and that the suspected (namely newly seen) ASN will be the same. In practice, this requirement may be relaxed and allow a few outliers. We identify all the APs with the same suspected ASN as a single event.

Table II summarizes the result statistics for the experiments we conducted for several arbitrary dates and times (see Table I). We choose times at different parts of the day and from different years. The most impressive aspect of the table is that all metrics manage to identify up to a few tens of suspected events from an original batch of over 10,000 APs with route change. *PAPES* and *Valley Diff* stand out with 2-16 events for each experiment. This is already a reasonable number to be handled by a response team.

To prevent repeated flagging of the same event, we filter out route changes that are caused by ASNs that were already flagged in the previous time slot. In continuous deployment, these APs have already been examined in the preceding analysis. We also remove events based on geo-filtering, where specific countries appear as both the hijack victims and attackers: Brazil, Iran, and Bangladesh. In these countries, ASes do not seem to obey the common BGP concatenation rules (aka valley-free routing [10]) and have complex relationships that

AP2Vec does not manage to learn. As a result, our algorithm is blind to local hijack attacks on ASes in these countries. We use [52] to obtain the owner-country of each ASN.

In the following sections, we will compare the detection rate and the false alarm rate of the five metrics and show that only *PAPES* achieves the right balance.

B. Results on Ground Truth Events

Table III summarizes the experiments conducted on ground-truth past events. For each event, we run our method five times, each time using a different metric. We mark with bold values of metrics that detect the event. *Roles ED* achieved the best result, successfully detecting 12 out of 13 events. *F-Roles ED* detected 11 out of 13. Both *PAPES* and *Local Hegemony Sim* detected 10 out of 13, and *Valley Diff* detected only 5 large-scale events and missed all the events that affected a few APs. However, as Table II shows, *PAPES* flags a significantly smaller number of incidents in comparison to the other methods: ~25% relative to *Roles ED* and 20-50% relative to *Local Hegemony Sim*, which reduces the load on the security management team.

Analyzing the cause of the misdetections made by *PAPES*, we found that in the *Iran Leaks* case, the algorithm highlights 13 of the 14 changed paths. The unflagged path has a similarity score of 0.286, which is only 0.012 higher than the threshold. Therefore by choosing a slightly lax condition the allows a few paths, which do not bit the threshold, we can improve the detection rate. In the *YouTube Hijacking* case, there were 33 changed paths. However, the RV record captured the attack when YouTube had already begun changing its announcements to mitigate the attack. Therefore, only 24 out of the 33 changed paths contain the attacker (Pakistan Telecom), while the rest contain AS36561 (YouTube).

TABLE IV
SUMMARY OF EXPERIMENTS ON ARBITRARY RECENT RECORDS

Record Time	APs w/ Changed Paths	Events	Affected APs	Suspected ASes	Post Geo-filtering	TP - Malicious	TP - Non-Malicious	FP - Same Countries	FP - Others	Siblings
2020-10-18 04:00	8,782	5	17	15	3	0	2	0	1	0
2020-10-18 16:00	10,208	5	48	16	3	1	1	0	1	0
2020-10-21 04:00	20,391	7	25	17	6	0	1	4	0	1
2020-10-21 16:00	14,957	14	71	24	11	1	1	4	3	2
2020-11-18 04:00	90,780	1	2	1	1	0	0	1	0	0
2020-11-18 16:00	38,023	13	65	22	8	0	2	3	3	0

While *PAPES* identify ground-truth events with high success rates and flags only a few candidate events, it is left to study the nature of the events flagged at random times by this method.

C. Evaluation of Recent Records

In this section, we analyze the events that were flagged by *PAPES* in recent time slots. Table IV summarizes our results based on manual analysis of all flagged records during six time-slots. For each analysis, we present the number of events, affected APs, and the number of suspected ASNs. Note that the results in the column “events” are post-filtering of events with ASNs that were part of events in the previous time slot; namely, we only present new events.

As we already mentioned, we filter events where both the AP and the suspected ASN hijacker are from the same country, for Brazil, Iran, and Bangladesh. This leaves us with 1-11 events per time slot. We manually analyzed each of these events and tabulated the results. We used the following categories: True positives (**TP - Malicious**), which are events that are suspected as hijack events; **TP Non-Malicious** which can be due to origin-change, traffic shifting to DDoS mitigators, or usage of routing optimizers (such as Asympto Networks AS39533). A few cases are due to siblings, and the rest are false positives, which are also divided into ones confined to a single country. Note that siblings and the FP that are in the same country can be filtered automatically, which will reduce the maximum number of events per time slot to five. Next, we describe a few of these events.

The true positive event of 21 Oct. 16:00 is due to an origin change for several APs that lasted between 14:10 and 16:50 (UTC). During this time, AS203, which belongs to Level3, appeared as the origin of 130 APs that belong to 13 ASNs. It is not clear if the announcements are due to a mistake at Level3 or if someone else used their ASN for a hijack attack. Note that while there are many origin changes, this one was flagged since AS203 is of a different type. We found a traceroute¹ measurement to one of the hijacked APs that did not reach the destination, while traceroutes from a day before and after the event reached it.

The true positive event of 18 Oct. 16:00 saw routes to 92 APs that belong to Hostinger International (AS47583), a cloud provider with a global footprint and offices in Cyprus and Lithuania, routed through a chain of four Indonesian ASNs: 136106, 38525, 9340, 7632. Our analysis identified

¹All the traceroute measurement we used for validating events come from an anonymized database

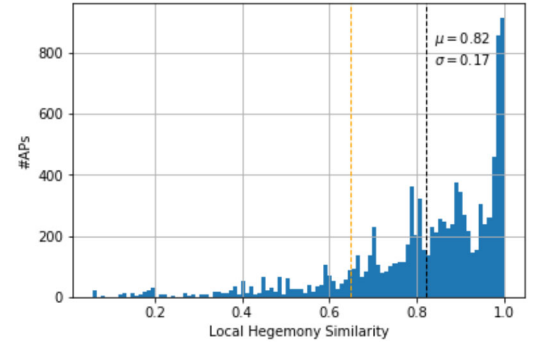


Fig. 8. Histogram of local hegemony similarities.

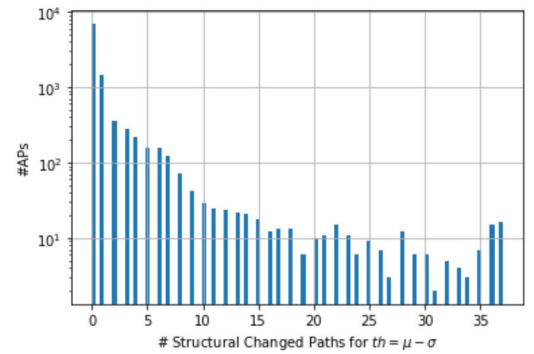
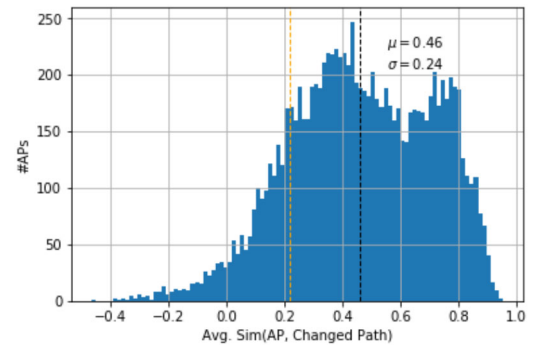


Fig. 9. Histograms of: (left) Avg. similarity between each AP and its changed paths, and (right) number of structural changed paths for each AP, where a structural changed path is determined by a similarity value lower than a threshold represented by the orange dashed line in the (left) graph.

three APs as problematic: 156.67.208.0/20 in Singapore and 195.110.58.0/23 and 31.170.164.0/23 in Great Britain. Additional 21 APs had multiple (at least 10) routes with functional changes. Most of them are also in Singapore and a few more in Great Britain. The APs in Great Britain are deflected across thousands of kilometers. To validate the deflection, we found a traceroute to 56.67.208.1 from Indonesia on Oct 18th

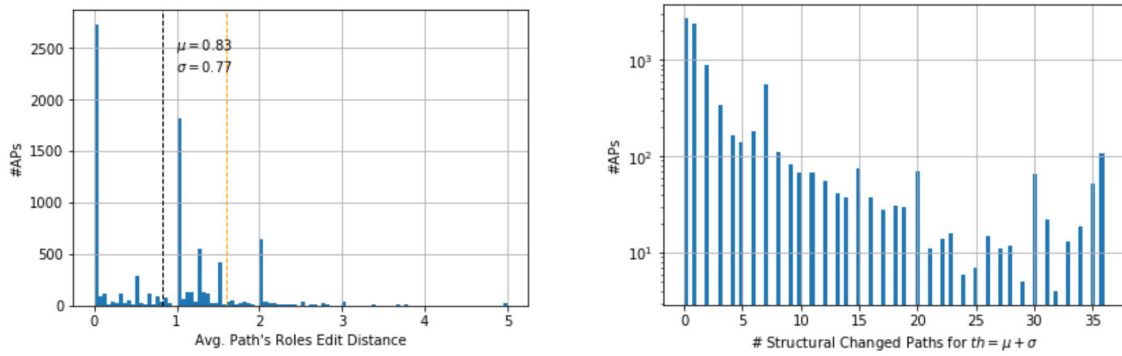


Fig. 10. Histograms of: (left) Avg. path's roles edit-distances using ASN-vectors-based roles, and (right) number of structural changed paths for each AP, where a structural changed path is determined by an edit-distance higher than a threshold represented by the orange dashed line in the (left) graph.

17:11 UTC that is crossing the Singapore Strait three times ID-SG-ID-SG, 24 hours earlier, the route is crossing the Strait only once.

An example of a route role change due to usage of a DDoS mitigator appears on 21 Oct. 4:00 (UTC), where the route to a portion of the AP 199.8.104.0/22 that belongs to AS1767 (Indiana Higher Education) was mitigated by SecurityDam (AS198949), an Israeli DDoS mitigator, through Internet 2.

Finally, we want to test whether the system may generate a wave of alerts that may overwhelm the human operators. For this purpose, we analyze the time periods before and after the two cases where the number of incidents flagged by *PAPES* was the largest, namely 21 Oct. 16:00 and 18 Nov. 16:00 (see Table IV). For both dates, the slots of 14:00 had only 4 incidents after geo-filtering. On each date, one of the incidents was a TP (Malicious). The slots of 18:00 had more incidents, 4 and 10 after geo-filtering, and only 1 and 3 incidents after removing same-country incidents. Out of these, there were 2 incidents due to DDoS mitigators.

IX. DISCUSSION AND TECHNICAL CONSIDERATIONS

Internet ASN roles do not change frequently. Thus, we can use the same model for an extended period. Namely, one can perform the relatively expensive (see below) model training and role extraction once a week or even once a month.

For a single CPU Windows server 2008 running on an Intel Xeon 2GHz, it took the AP2Vec training about 2.5-3.5 hours and the role extraction 1.5-2 hours. The offline analysis of an RV file requires about 4-5 minutes for importing the RV data.

To build an online hijack detection service for the entire Internet, we will upload the BGP announcements as they come. The various metric calculations can be done periodically for each AP. For a 2-hour RouteViews file with roughly 30 million routes, the local hegemony similarity and *PAPES* calculations took about 5 minutes, the global hegemony valley difference took about 1 minute, and both role edit-distances took a few seconds to calculate.

We can run such a system in real-time, where the system is fed from a source such as BGPstream [53] and once it collects a sufficient number of routes per origin (say 5), it performs the analysis for this collection. One can expect slightly higher false alarm rates since we cannot filter based on the entire group of routes.

While this paper concentrate on building a global scale alert system by identifying significant route deflection attacks, our techniques can be adapted for monitoring only a small portion of the Internet, namely a single AS. To do this, one can train the system using a large collection of routes, as we did in this paper, and then fine-tune the embedding for the ASN we are interested in. Using the embedding, one can simply look for cases with changes in the roles along the path.

X. CONCLUSION

We introduced a novel approach for BGP hijacking detection (by classifying BGP routes as hijacked or benign) using an unsupervised deep learning approach. It is based on the observation that in a hijack attack, the sequence of ASN functions along the path changes, so our method detects this functionality change. We showed that we could detect past hijack attacks with high probability, and at the same time, maintain a low amount of flagged events.

In this work, we emphasized the practicality of the system since the success of such a system will be measured by the trust of the human team that handles its alerts. Our manual analysis of the events flagged by the system showed that not only are they a few, but they represent interesting routing changes even though not necessarily malicious. Overall in 6 time-slots, our system flagged 32 events, which can be automatically filtered by geography and siblings to only 17 events, where only 8 of them were false alarms. The 9 TP events are divided into 2 possibly malicious events and 7 events that are due to benign hijack events by DDoS mitigators and route optimizers. These benign events can be filtered by maintaining a list of ASNs, which are benign hijackers. When analyzing time slots adjacent to the ones with the most incidents, we see significantly fewer incidents. Namely, there are no waves of many incidents reported by the system. Overall, when compared to other cyber detection systems our system achieves similar results to state-of-the-art approaches but with a fraction of false-positive alerts. Gao [10] and Arthemis [9] produce 2-3 orders of magnitude more alerts on a global basis (see introduction).

While practical, our work also opens the door to a better understanding of ASN functional roles and methods to reveal them. We studied several similarity metrics, but we believe that there is more to uncover. Hopefully, understanding roles can

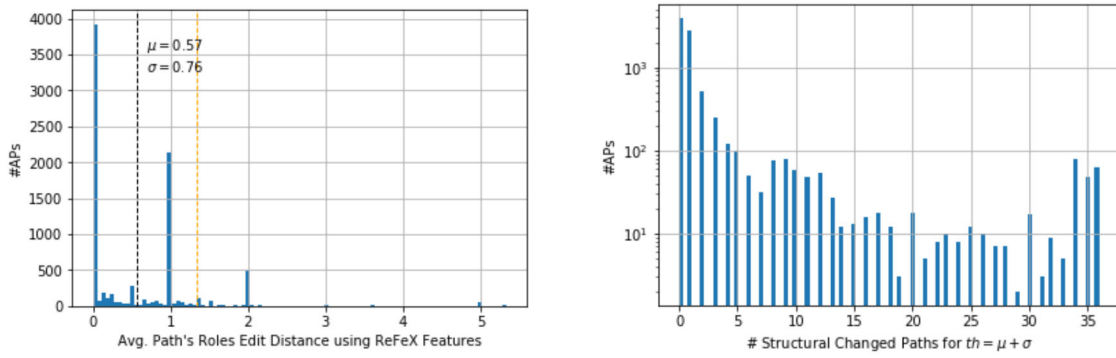


Fig. 11. Histograms of: (left) Avg. path's roles edit-distances using *ReFeX* features-based roles, and (right) number of structural changed paths for each AP, where a structural changed path is determined by an edit-distance higher than a threshold represented by the orange dashed line in the (left) graph.

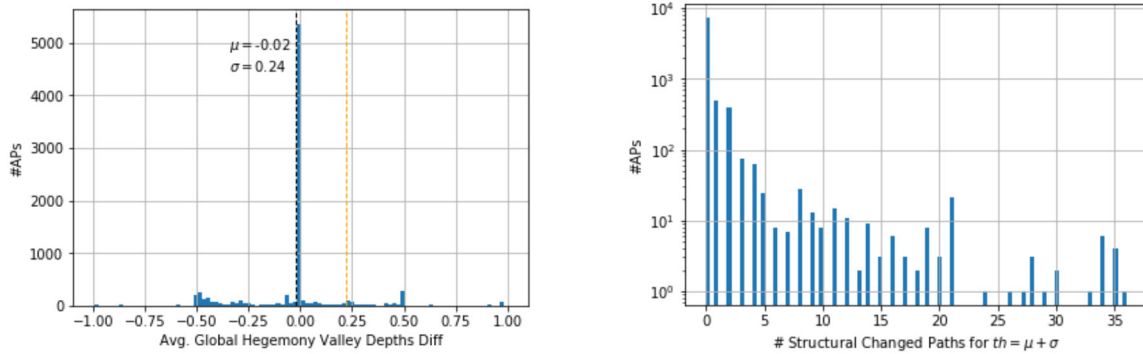


Fig. 12. Histograms of: (left) Avg. AS global hegemony valley depth differences, and (right) number of structural changed paths for each AP, where a structural changed path is determined by a difference value higher than a threshold represented by the orange dashed line in the (left) graph.

lead to a better understanding of the engineering and economic aspects of the Internet ecosystem.

APPENDIX METRICS ANALYSIS

Figures 8, 9, 10, 11, 12 present the parameters analysis for the first phase of our unsupervised method, i.e., defining thresholds for a significant structural change. For each metric, we generate a histogram, such that the Y-axis represents the percentage of APs for the specified metric value. Apart from the local hegemony similarity, which is calculated on the entire list of changed paths, for each AP, we calculate the average metric among its changed paths. We conduct the analysis based on the RV record from 2020-10-18 16:00. Notice that all the histograms include only APs that their routes were changed.

Based on the chosen thresholds for each metric (excluding local hegemony similarity) and for each AP, we count the number of structural-changed paths. For all the metrics we use, more than 80% of the APs with path changes have less than 5 paths with structural roles changes. The presented distributions are similar to other time slots.

REFERENCES

- [1] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, "A survey among network operators on BGP prefix hijacking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 1, pp. 64–69, Jan. 2018.
- [2] C. C. Demchak and Y. Shavitt, "China's Maxim—Leave no access point unexploited: The hidden story of china telecom's BGP hijacking," *Military Cyber Affairs*, vol. 3, no. 1, p. 7, Oct. 2018.
- [3] Y. Gilad, T. Hlavacek, A. Herzberg, M. Schapira, and H. Shulman, "Perfect is the enemy of good: Setting realistic goals for BGP security," in *Proc. 17th ACM Workshop Hot Topics Netw.*, 2018, pp. 57–63.
- [4] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "BGP hijacking classification," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2019, pp. 1–8.
- [5] G. Huston and R. Bush, "Securing BGP and SIDR," *IETF J.*, vol. 7, no. 1, pp. 8–13, 2011.
- [6] M. Lepinski and K. Sriram, "BGPSEC protocol specification," Internet Eng. Task Force, RFC 8205, 2017.
- [7] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are we there yet? On RPKI's deployment and security," in *Proc. NDSS*, Feb. 2017, pp. 1–15.
- [8] T. Chung *et al.*, "RPKI is coming of age: A longitudinal study of RPKI deployment and invalid route origins," in *Proc. Internet Meas. Conf.*, 2019, pp. 406–419.
- [9] P. Sermpezis *et al.*, "ARTEMIS: Neutralizing BGP hijacking within a minute," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2471–2486, Dec. 2018.
- [10] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, Dec. 2001.
- [11] K. Balu, M. L. Pardal, and M. Correia, "DARSHANA: Detecting route hijacking for communication confidentiality," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, 2016, pp. 52–59.
- [12] P. Moriano, R. Hill, and L. J. Camp, "Using bursty announcements for detecting BGP routing anomalies," *Comput. Netw.*, vol. 188, Apr. 2021, Art. no. 107835.
- [13] T. Shapira and Y. Shavitt, "A deep learning approach for IP hijack detection based on ASN embedding," in *Proc. Workshop Netw. Meets AI ML*, 2020, pp. 35–41.
- [14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2013, pp. 3111–3119.
- [15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

- [16] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and whisper: Security mechanisms for BGP," in *Proc. 1st Symp. Netw. Syst. Design Implement. (NSDI)*, 2004.
- [17] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: Improving BGP by cautiously adopting routes," in *Proc. ICNP*, 2006, pp. 290–299.
- [18] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A prefix hijack alert system," in *Proc. USENIX Security Symp.*, vol. 15, 2006, p. 11.
- [19] J. Qiu, L. Gao, S. Ranjan, and A. Nucci, "Detecting bogus BGP route information: Going beyond prefix hijacking," in *Proc. Security Privacy Commun. Netw. (SecureComm)*, 2007, pp. 381–390.
- [20] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting IP prefix hijacks in real-time," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 277–288, 2007.
- [21] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "ISPY: Detecting IP prefix hijacking on my own," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 327–338, 2008.
- [22] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 3–17.
- [23] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the Internet with argus," in *Proc. Internet Meas. Conf.*, 2012, pp. 15–28.
- [24] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. W. Biersack, "HEAP: Reliable assessment of BGP hijacking attacks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 6, pp. 1849–1861, Jun. 2016.
- [25] M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, "MS-LSTM: A multi-scale LSTM model for BGP anomaly detection," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, 2016, pp. 1–6.
- [26] Q. Ding, Z. Li, P. Batta, and L. Trajković, "Detecting BGP anomalies using machine learning techniques," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, 2016, pp. 3352–3355.
- [27] H. Alshamrani and B. Ghita, "IP prefix hijack detection using BGP connectivity monitoring," in *Proc. IEEE 17th Int. Conf. High Perform. Switch. Routing (HPSR)*, 2016, pp. 35–41.
- [28] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling BGP serial hijackers: Capturing persistent misbehavior in the global routing table," in *Proc. ACM Internet Meas. Conf. (IMC)*, 2019, pp. 420–434.
- [29] "Route Views Project." University of Oregon Advanced Network Technology Center. [Online]. Available: <http://www.routeviews.org/> (Accessed: Nov. 18, 2020).
- [30] R. Fontugne, A. Shah, and E. Aben, "The (thin) bridges of AS connectivity: Measuring dependency using as hegemony," in *Passive and Active Measurement*, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 216–227.
- [31] "Dyn Blog." Dyn. 2020. [Online]. Available: <https://hub.dyn.com/dyn-research>
- [32] "Ripe NCC Publications." Ripe NCC. 2020. [Online]. Available: <https://www.ripe.net/publications/>
- [33] "BGPMon Blog." BGPMon. 2020. [Online]. Available: <https://www.bgpmon.net/blog/>
- [34] "Apnic Blog." Apnic. 2020. [Online]. Available: <https://blog.apnic.net/>
- [35] "Oracle Internet Intelligence." Oracle. 2020. [Online]. Available: <https://blogs.oracle.com/internetintelligence/>
- [36] "Manrs Blog." Manrs. 2020. [Online]. Available: <https://www.manrs.org/news/>
- [37] "BGProtect Case Studies." BGProtect. 2020. [Online]. Available: <https://www.bgprotect.com/case-studies/>
- [38] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 701–710.
- [39] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, 1999.
- [40] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 71–74, Oct. 2005.
- [41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arxiv:1301.3781*.
- [42] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [43] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [44] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. LREC Workshop New Challenges NLP Frameworks*, May 2010, pp. 45–50.
- [45] H. Hyrö, "Explaining and extending the bit-parallel approximate string matching algorithm of myers," Dept. Comput. Inf. Sci., Univ. Tampere, Tampere, Finland, Rep. A-2001-10, 2001.
- [46] K. Henderson *et al.*, "Rolx: Structural role extraction & mining in large graphs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2012, pp. 1231–1239.
- [47] K. Henderson *et al.*, "It's who you know: Graph mining using recursive structural features," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2011, pp. 663–671.
- [48] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
- [49] T. Shapira and Y. Shavitt, "Unveiling the type of relationship between autonomous systems using deep learning," in *Proc. IEEE/IFIP NOMS Int. Workshop Anal. Netw. Service Manage. (AnNet)*, Apr. 2020, pp. 1–6.
- [50] "Hurricane Electric Internet Services." Hurricane Electric. 2020. [Online]. Available: <https://bgp.he.net/>
- [51] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.
- [52] "Caida as Rank." Oct. 2020. [Online]. Available: <http://as-rank.caida.org/>
- [53] C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti, "BGPStream: A software framework for live and historical BGP data analysis," in *Proc. ACM IMC*, 2016, pp. 429–444.



Tal Shapira (Graduate Student Member, IEEE) was born in Rishon-Lezion, Israel, in 1991. He received the B.Sc. degree in physics and minor in mathematics from the Hebrew University of Jerusalem, Jerusalem, Israel, in 2012, and the M.Sc. degree (*magna cum laude*) in electrical engineering and the Ph.D. degree from Tel-Aviv University, Tel-Aviv, Israel, in 2018 and 2022, respectively.

He has been a Postdoctoral Fellow with the School of Computer Science, Reichman University, Herzliya, Israel, since 2022. From 2009 to 2012, he served as a Cadet with the Talpiot elite Israeli Defense Forces program. From 2013 to 2015, he served with the Israeli Prime Minister Office as an RF and Antennas Engineer, from 2015 to 2018 as a Data Scientist and a Data Science Team Leader, and in the last two years of his military service as the Head of a Cybersecurity R&D Group. After graduation, he worked as the Head of Deep Learning and Computer Vision Algorithms Group in an automotive startup called Guardian Optical-Technologies. In 2020, he co-founded RecoLabs Inc., where he serves as the Chief Scientist. His research focuses on deep learning, computer networks, and cybersecurity.



Yuval Shavitt (Senior Member, IEEE) received the B.Sc. degree (*cum laude*) in computer engineering, the M.Sc. degree in electrical engineering, and the D.Sc. degree from Technion, Israel Institute of Technology, Haifa, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served with the Israel Defense Forces first as a System Engineer and the last two years as a Software Engineering Team Leader. After graduation, he spent a year as a Postdoctoral Fellow with the Department of Computer Science, Johns Hopkins University, Baltimore, MD. From 1997 to 2001, he was a Member of Technical Staff with the Networking Research Laboratory, Bell Labs., Lucent Technologies, Holmdel, NJ. Since October 2000, he has been a Faculty Member with the School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel. His recent research focuses on Internet measurements and deep learning solutions for various networking problems.

Prof. Shavitt served as a TPC member for many networking conferences and on the Executive Committee of INFOCOM 2000, 2002, and 2003. He was the TCP Co-Chair of TMA 2011, a Keynote Speaker of PAM 2012, an Editor of *Computer Networks* from 2003 to 2004, and served as a Guest Editor for *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS* and *JWWW*. In 2014, he co-founded BGProtect Ltd., a company that monitors the Internet for IP hijack attacks, where he serves as the CTO.