DeepGFL: Deep Feature Learning via Graph for Attack Detection on Flow-based Network Traffic

Yepeng Yao[1,2], Liya Su[1,2], Zhigang Lu[1,2†]
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
Email: {yaoyepeng, suliya, luzhigang}@iie.ac.cn

***Abstract*—The ability to mine structurally complex and latent relationship among network flows has become the focus of many initiatives. Learning graph representation for network attack detection has become a critical issue which is an across-network machine learning task. However, the challenge of effectively representing graph for network traffic is unmet yet, especially for detecting various threat patterns which is modeled as attributed graph. In the same time, existing methods could not capture higher-order subgraph structures. For these reasons, we propose a new way to model network graph called Deep Graph Feature Learning (DeepGFL) for network attack detection to solve this problem. DeepGFL is a framework studying deep features from attributed network flow graph. We automatically generalize higher-order features from raw features obtained from attributed graphs and then implement network attack detection. We evaluate the proposed framework with raw features threat detection on a real world datasets. Experimental results show that DeepGFL is more effective, more accurate and more space efficient for network attack detection.**

Keywords—graph representation, deep graph feature learning, feature function, network attack detection, network security

I. INTRODUCTION

Large scale IT infrastructures are threatened by an ever-growing number of different threats. Cyber-attacks usually hide communication data in massive legitimate network traffic to evade the detection of security devices and achieve the purpose of long-term latent and information theft. The current existing cyber-attacks, such as the malware, botnet and APT, can be classified as such type attacks. Even though network level cyber-attack detection is widely studied by the community as the first line of defense against cyber threats, these attacks are still hard to detect because their network activities are subtle and do not cause target network sharp fluctuation or disruption in contrast to other attacks [1]. The ability to mine structurally complex and latent relationship among network flows has become the focus of many initiatives, ranging from malware traffic analysis to network intrusion detection. These applications often represent the underlying network traffic as a graph for various reasons, but most importantly for the computational efficiency and scalability that graph techniques enable. Due to their wide usages, many interesting graph problems are extensively studied, such as node embedding and graph kernel.

However, most of these past works have focused on node features. These node features didn't provide enough useful representation of the network flow graph. In addition, existing methods are also unable to leverage attributes, methods like node embedding are not suitable for attributed graph. For example, node2vec can't treat with network flows and usually lose important connected information.

Learning a useful graph representation has become a critical issue in network attack detection as an across-network machine learning tasks. In this work, we consider the network attack detection problem for attributed network flow graphs, which contain different types of hosts and communication traffic. The focus of our proposed approach is to present deep graph-based method to uncover attacks in network traffic containing possible network attack activities. Recall that, raw features are not representable enough.

In this paper, we present a deep graph feature learning framework called DeepGFL, which overcomes many of the above limitations, to extract higher-order features in the context of network security. Our goal is to derive higher-order network flow features from lower-order ones forming a hierarchical graph representation where each layer consists of features of increasingly higher orders.

The main contributions of our work are as follows:

- We design a model for extracting higher-order features in the context of network security aiming at detecting network attacks.
- We propose a graph-based feature learning algorithm to represent the network flow relationships on hosts, and perform a feature evaluation routine to choose the important features exposing the different patterns between benign and attack network flows.
- We use insights from our measurements to build an attack detection prototype, using deep graph features learning from network flow graph, and evaluate it on a real world dataset.

To the best of our knowledge, this paper presents the first proposal of models, algorithms and analyzers integrated in a real attack detection prototype, which may be applicable to attributed network flow graphs with high detection accuracy and space efficiency.

The rest of this paper is organized as follows. Section II gives an introduction of DeepGFL framework, and points out that the purposes of this paper: firstly, extracting higher-order features from the raw network flow features; furthermore, learning and extracting the hierarchical graph representation

This work is supported by Natural Science Foundation of China (No. 61702508, No.61602470, No.61572481). This work is also partially supported by Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences and Beijing Key Laboratory of Network Security and Protection Technology.

† Corresponding author.

from raw flow features and higher-order features; finally evaluating and choosing the important features. Section III presents the main algorithm towards deep graph feature learning in detail. Section IV describes the prototype of deep graph feature for attack detection. Section V presents an experimental evaluation on a real world dataset consisting of about 3 million network flows. Section VI gives an overview of the related work. Section VII concludes the paper and discusses the future work.

II. Design

To achieve the purpose of long-term latency, traffic patterns of some attacks cannot be characterized by only one flow, and instead require an aggregation of the related flows' information. For the detection of these attack scenarios, we provide brief explanations of the framework of DeepGFL on flow-based network traffic in this section.

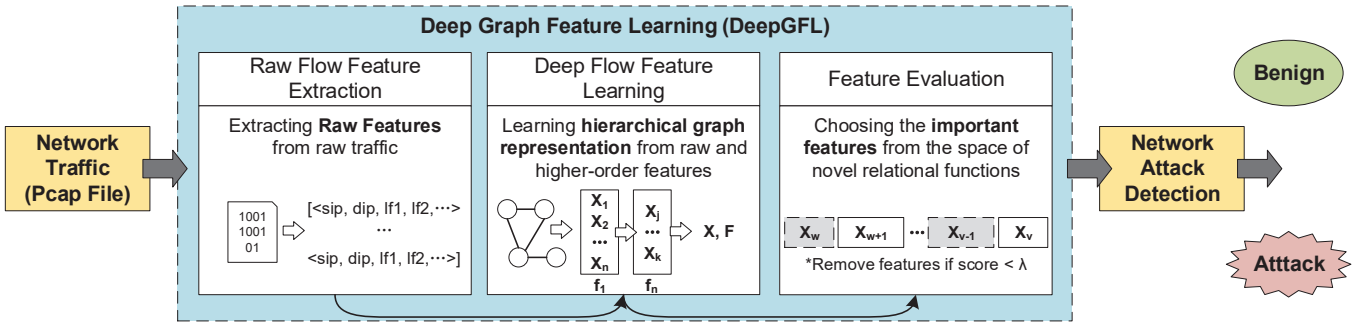


Fig. 1. Overview of the deep graph feature learning (DeepGFL).

A. Framework Overview

First, we present the overview of Deep Graph Feature Learning (DeepGFL) framework for attack detection. Intuitively, the DeepGFL should be a process in which learns the deep graph features of network flows and then detects attack in network flows.

In Fig. 1 we give an overview of the complete processing chain from deriving raw flow features to obtaining the threat detection result. First we describe extracting raw flow features, learning deep graph features, feature pruning and feature evaluation. And then describe the threat detection system.

B. Raw flow Features

The first step of DeepGFL is to derive a set of raw flow features using the network traffic flow features and network flow graph topology attributes. It should be noted that Deep GFL can use an arbitrary set of raw features from network flows, and thus it is not limited to the raw features discussed below. In the same time, the better the raw features, the better DeepGFL learns the deep graph features. For extracting the network traffic raw features, we adopted the CIC FlowMeter [8] tool, which is a flow based feature extractor and can extract 80 features from raw network traffic.

In the second step, to find the best feature set for detecting each attack from 80 extracted features, we adopt Random Forest Regressor class which calculates the importance of each feature in the whole dataset. It achieves the final result by multiplying the average standardized mean value of each feature split on each class, with the corresponding feature importance's value. Ref [8] shows the selected features and corresponding weight of each section.

As [8] shows, eight raw features are very important for the majority of attack scenarios and common enough for other feature extract tools include the following: *flow duration, total forward packets, total backward packets, total length of forward packets, total length of backward packets, flow bytes per seconds, flow packets per seconds, down/up ratio*.

For the next step of our analysis, we use the eight raw features mentioned above for simplifying the experiment and getting the prominent experimental result.

Observe that DeepGFL naturally supports other raw feature sets and many graph properties including efficient/linear-time properties such as PageRank. Moreover, fast approximation methods with provable bounds can also be used to derive features such as the largest clique centered at the neighborhood of each graph element (node, edge) in G.

C. Deep Graph Features Learning

Given a graph $G = <V, E>$, we first decompose G into its smaller subgraph components [7] and use these features to learn deep features. This work derives deep features by using feature functions to all node or edge features within subgraphs. Importantly, DeepGFL handles multiple types of subgraph sizes and features including subgraphs that are directed/undirected, typed/heterogeneous, and/or temporal. Furthermore, one can also derive such subgraph features efficiently by leveraging fast and accurate subgraph estimation methods (e.g.,[5][6]).

DeepGFL has a primary advantage that it can handle the attributed graphs. We discuss how to learn a node or edge feature-based representation given an initial set of node or edge attributes. For learning a node representation, given G and an initial set of related node attributes, we simply derive node features by applying the set of feature functions to each node attribute. Conversely, learning an edge representation, given G and an initial set of related edge attributes, we derive edge features by applying each feature function $o \in O$ to the edges

of the nodes at either end of the edge. Then the input attributes match the type of graph element(node, edge) and simply append to the feature matrix X.

TABLE I. EXAMPLES OF FEATURE FUNCTIONS

Functions	Definition	Representation
Hadamard	$O\langle S,x\rangle = \prod_{s_j \in S} x_j$	*Mul*
Mean	$O\langle S,x\rangle = \frac{1}{\lvert S\rvert}\sum_{s_j \in S} x_j$	*Mean*
Sum	$O\langle S,x\rangle = \sum_{s_j \in S} x_j$	*Sum*
Weight.Lp	$O\langle S,x\rangle = \sum_{s_j \in S} \lvert x_i - x_j\rvert^p$	*Diff*

The feature functions is formulated as $O \in \{o_1,...,o_n\}$. Some sample feature functions are provided in TABLE I. Ref [3] provides a wide variety of other useful feature functions. Composing several feature functions could derive a new feature function. The depth of the DeepGFL depends on three components including: (1) the initial raw features derived from network flows; (2) the set of feature functions; (3) the number of times of using feature function operators. Under this formulation, each feature vector x_0 from x can be written as a composition of feature functions applied over the raw features. And even more complex feature functions are easily expressed as the compositions of different feature functions. Moreover, DeepGFL learns higher-order features based on a set of initial raw graph features. Although the learning processing is similar as Convolutional Neural Networks (CNNs) [4], DeepGFL captures higher-order features from lower-order features in each successive layer with understandable operators.

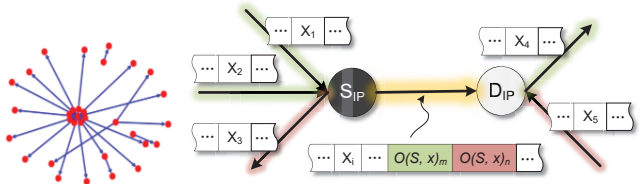


Fig. 2. Network flow graph with deep graph representation. (a) a subgraph example for 100 flows; (b) An intuitive example for an edge $e = (S_{IP}, D_{IP})$ and a feature function $o \in O$

A general and flexible deep feature learning framework for DeepGFL is given in Alg. 1, Section III. DeepGFL begins with a set of raw features derived from network flows and uses these as a basis for learning deeper and more discriminative features by increasing complexity. The framework proceeds to learn a hierarchical graph representation where each successive layer represents increasingly deeper higher-order (edge/node) graph functions (due to composition): $F_1 < F_2 < F_3 ... < F_n$ s.t. if $i < j$ then F_j is said to be deeper than F_i. In particular, the feature $F_2, F_3,...,F_n$ are learned as follows (Alg. 1): First, we derive the feature layer F_n by searching over the space of graph functions that arise from applying the feature function O to each of the novel features $f_i \in F_{n-1}$ learned in the previous layer. Further, an intuitive example is provided in Fig. 2.

D. Feature Pruning

The resulting features in layer n are then evaluated. The feature evaluation routine chooses the important features (feature functions) at each layer n from the space of novel feature functions (at depth n) constructed by composing the feature function learned in the previous layer $n-1$. Notice that DeepGFL is flexible for the feature evaluation routine changing and fine-tuned for different network flow graphs.

Feature evaluation function is used to select the subset of important features. Features are selected as follows: First, we calculated the evaluation score of each feature. Next, compare the deeper feature score with the threshold value and the raw feature score. If the deeper feature score is larger than the both, then discards the raw feature. Recall that the feature evaluation routine described above is completely interchangeable by replacing the evaluation function in DeepGFL framework. The detail of the feature evaluation is shown in Section III.

After pruning the feature layer F_n, the discarded features are removed from X and DeepGFL updates the set of features by setting $F \leftarrow F \cup F_n$. Next, it increments n and sets $F_n \leftarrow \varnothing$. Finally, we check for convergence: if the stopping criterion is not satisfied, then DeepGFL tries to learn an additional feature layer. In contrast to node embedding methods that output only a node feature matrix X, DeepGFL also outputs the feature functions (operators) $O_d \in \{o_1,...,o_n\}$ where each $o_i \in O_d$ is a learned feature function of depth d for the *i-th* feature vector x_i. Maintaining the feature functions are important for transferring the features, but also for interpreting them.

III. DEEP GRAPH FEATURE LEARNING ALGORITHM

In this section, we describe the deep graph feature learning algorithm to perform higher-order representations.

Algorithm 1 The DeepGFL framework for learning deep graph representations from network flow graphs

Require: a network flow set *NF*; a set of feature functions $O \in \{o_1,...,o_n\}$; an upper bound *max* on the number of flows to construct the flow graph

Repeat

> Construct flow graph $G = <V, E>$ by less than *max* network flows from *NF*, then add the raw flow feature vectors to X and definitions to F_1, set $F \leftarrow F_1$.
>
> Search the space of features defined by applying feature functions to the previous layer features.
>
> Evaluate the features and feature functions with the evaluation method to select a subset.

Until no new features emerge or the max number of flows is handled

Return X and the set of feature functions O

A. Feature Functions for Network Flows

First we decompose the graph into its smaller subgraph components that conclude in/out/total graph elements. DeepGFL derives higher-order features by calculating in/out/total node or edge features within subgraphs. Importantly, DeepGFL handles multiple types of subgraphs and features including subgraphs that are directed/undirected, typed/heterogeneous, and/or temporal. Furthermore, as for large network flow graph one can derive such subgraph featurest efficiently by leveraging fast subgraph estimation methods.

The set of feature functions $O \in \{o_1, ..., o_n\}$ are composed for calculating the higher-order features. For edge feature learning we derive edge deeper features for each edge $(v,u) \in E$ as follows:

$$\left[f_v^+ \circ f_u^+, f_v^- \circ f_u^-, f_v^- \circ f_u^+, f_v^+ \circ f_u^-, f_v \circ f_u \right] \quad (1)$$

where $f_v = f_v^+ \circ f_v^-$ and f_v^+, f_v^-, f_v denote the out/in/total edge features of *v*.

However, our work focuses on learning directed graph features and feature functions for network flows. We propose a more appropriate way for network flows. As we all known, the in edge features of source node and out edge features of destination node are more representative for learning, so we use $f_v^+ \circ f_u^-$ to learn the higher-order features for network flows.

Finally, check for convergence, DeepGFL tries to learn an additional feature layer until satisfied the stopping criterion. Learning from experiment, when the layer is larger than 5 the features are usually convergent so we set 6 layer number as the stopping criterion.

B. Feature Evaluation for Network Flows

In this paper, we want to estimate the correlation between the lower-order features and higher-order features. So we use the Correlation Feature Selection (CFS) measure proposed by [2]. The CFS measure considers correlation between a feature and a class and inter-correlation between features in the meantime. This measure finds the globally optimal subset of relevant features which is used successfully in test theory.

We use the evaluation method proposed by [2], and find the subset of features which have the maximum value of $Merit_s(k)$.

$$Merit_s(k) = \frac{r_{cf_1} + r_{cf_2} + ... + r_{cf_k}}{\sqrt{k + 2\left(r_{f_1f_2} + r_{f_1f_3} + ... + r_{f_1f_k}\right)}} \quad (2)$$

Recall the feature evaluation routine described above is completely interchangeable by replacing the evaluation function in DeepGFL framework such as Principal Component Analysis(PCA), Decision Trees and the Pegasos method [14].

IV. Attack Detection on Network Flows

In the previous section we identify several features that indicate network attack activity. None of these features, taken individually, are sufficient for detecting most of the attack in our data set.

In this section, once the deep graph features for the flow graphs has been computed, we introduce the first attempt to use the flow-based deep graph features for attack detection purposes. We therefore build an attack detection prototype that employs supervised machine learning techniques for automatically selecting the best combination of features to separate the benign and attack network flows. Specifically, we train two state-of-the-art classifiers to perform the attack detection respectively, namely Decision Tree and Random-Forests. One important advantage of Random-Forests is that the variance of the model decreases as the number of trees in the forest increases, whereas the bias remains the same.

V. Evaluation

In our experimental study, we conducted experiments to test the effectiveness and efficiency of the proposed framework in learning deep graph features over network flow graphs.

A. Experimental Settings

1) Datasets: We choose CIC-IDS-2017 dataset [8] as our experiment dataset, which contains 2,830,743 flows in total. This is one of the newest labeled intrusion detection dataset, which covers all the eleven necessary criteria with common updated attacks such as DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port Scan and Botnet. It was not divided by the provider into training and test datasets; therefore, we divide it into training and test data sets using a ratio of 40% to 60%. We also removed Heartbleed, Infiltration and SQL Injection because the traffic data of these three attack scenarios has only less than 100 attack flows, respectively.

2) Testbed: We implemented a proof-of-concept version of our deep graph feature learning framework (see Section III and Section IV). Python3, NetworkX1.9 and Sklearn0.19, which are run on the Ubuntu 16.04 64-bit OS, are used as the software frameworks. The server is a DELL R720 with 16 CPU cores and 128 GB of memory.

3) Metrics: For evaluation, we report the precision, recall, *F1* score and AUC (area under ROC curve) value for each network attack scenario.

B. Experimental Design

In our experiment, we construct flow graphs of every 1000 flows for the calculation of deep graph features. The in-edge features of source host of flows and out-edge features of destination host of flows have been used to learn higher-order features. We structured our experiments in three parts:

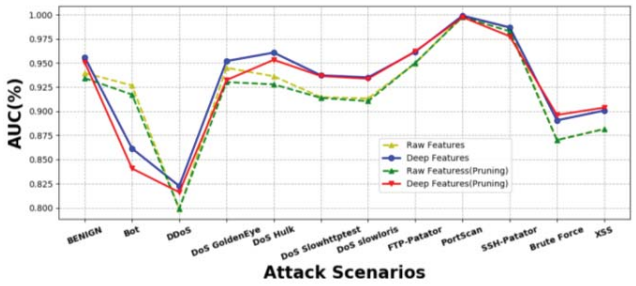
- Evaluate and compare the effectiveness of the raw features, deep graph features, raw features after pruning and deep graph features after pruning, then determine whether our prototype is able to detect network attacks under different detection methods.

- Evaluate and compare the performance of four different feature functions, respectively.
- Estimate and analysis the space efficiency of DeepGFL.

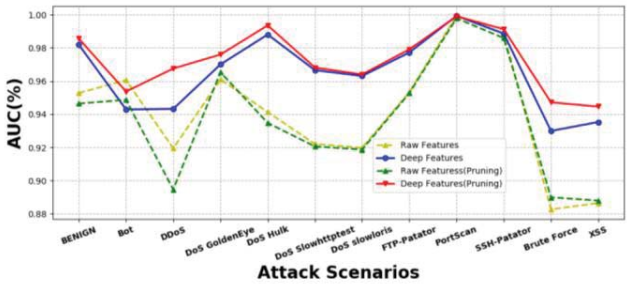
C. Results

1) Effectiveness of raw features, deep graph features and deep graph features after pruning

Results have shown the effectiveness of DeepGFL for attack classification in Fig. 3 and the overall precision, recall and *F1* are presented in TABLE II.



(a) Decision Tree classifier



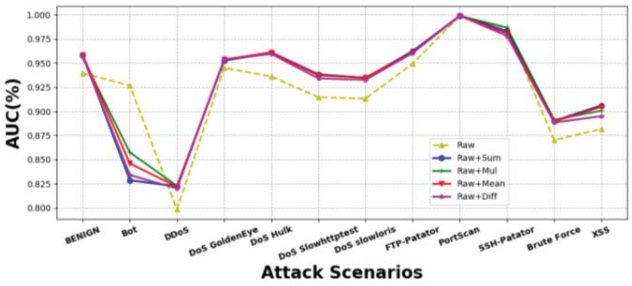
(b) Random-Forests classifier

Fig. 3. Classification accuracy over raw features, deep features, raw features after pruning and deep features after pruning. After pruning, only six raw features were left, that were flow packets per secends, flow bytes per secends, flow duration, total length of backward packets, total length of forward packets and total forward packets.

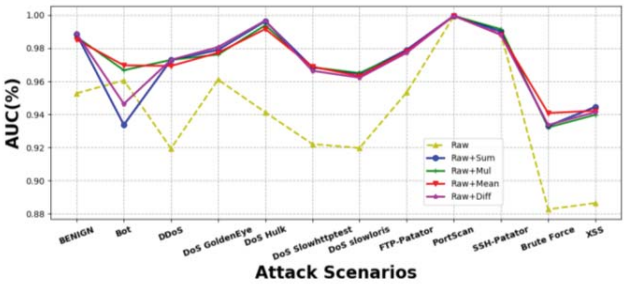
We focused on four different feature collections that may be representative. For each attack, our approach works clearly better than other approaches in attack detection. And the worst performing over deep features after pruning is always higher than 92% when adopts Random-Forests as the classifier.

2) Performance with different feature functions

Fig. 4 shows the improvement of attack detection performance on the whole dataset when choosing different feature functions. We make the following observations: for Random-Forests classifier, the *Mul* and *Mean* feature functions give best performance on average across all attack scenarios with an AUC of higher than 93% over all scenarios.



(a) Decision Tree classifier



(b) Random-Forests classifier

Fig. 4. Variation of AUC against attack scenarios. From this figure, it is clear that the performance of DeepGFL is efficient for improving classification accuracy over four feature functions.

TABLE II. PERFORMANCES OF RANDOM-FORESTS CLASSIFIER OVER RAW FEATURES, DEEP GRAPH FEATURES, RAW FEATURES AFTER PRUNING AND DEEP GRAPH FEATURES AFTER PRUNING

Attack Scenarios	Raw Features			Raw Features after Pruning			Deep Features			Deep Features after Pruning		
	Precision	*Recall*	*F1*	*Precision*	*Recall*	*F1*	*Precision*	*Recall*	*F1*	*Precision*	*Recall*	*F1*
BENIGN	0.91603	0.98460	0.94908	0.90998	0.98948	0.94807	0.97092	0.98869	0.97973	**0.97297**	**0.99024**	**0.98153**
Bot	0.99726	0.31434	0.47800	0.94886	0.28843	0.44238	0.92069	0.23057	0.36878	0.91391	0.23834	0.37808
DDoS	0.40814	0.08332	0.13839	0.40925	0.10042	0.16126	0.74157	0.29342	0.42047	**0.75672**	**0.30240**	**0.43212**
DoS GoldenEye	0.97913	0.56806	0.71899	0.96417	0.60391	0.74265	0.97507	0.47376	0.63769	0.96306	**0.58098**	**0.72475**
DoS Hulk	0.73086	0.35256	0.47566	0.78313	0.27028	0.40187	0.91114	0.94098	0.92582	**0.92629**	**0.95526**	**0.94055**
DoS Slowhttptest	0.94971	0.14462	0.25102	0.87708	0.15552	0.26420	0.87679	0.14462	0.24829	**0.88235**	**0.15906**	**0.26953**
DoS slowloris	0.98563	0.13825	0.24249	0.95501	0.13450	0.23580	0.98069	0.13162	0.23210	**0.98768**	**0.13854**	**0.24299**
FTP-Patator	0.99176	0.48442	0.65091	0.99178	0.48570	0.65206	**0.99564**	0.48337	0.65078	0.99476	0.48294	0.65021
PortScan	0.99219	0.99557	0.99388	0.99150	0.99386	0.99268	0.99194	0.99339	0.99267	**0.99305**	**0.99557**	**0.99431**
SSH-Patator	0.99769	0.49697	0.66346	0.99475	0.49178	0.65818	0.97364	0.46843	0.63254	**0.98904**	**0.46843**	**0.63576**
Brute Force	1.00000	0.04075	0.07831	1.00000	0.03971	0.07638	0.95122	0.04075	0.07816	**1.00000**	**0.04284**	**0.08216**
XSS	1.00000	0.02597	0.05063	1.00000	0.02597	0.05063	1.00000	0.02597	0.05063	**1.00000**	**0.02597**	**0.05063**

3) Space efficiency

In all cases, the flow representations learned by DeepGFL are extremely sparse and significantly more space-efficient than raw features as observed in experiment. Strikingly, DeepGFL uses only a fraction of the feature space required by existing methods and records the feature functions. In the first

experiment, the feature pruning process reduced 2 raw features, namely total backward packets and down/up ratio - a significant reduction in space by a factor of 25%.

VI. Related Work

In this section, we highlight how DeepGFL differs from related works.

A. Graph Feature Based Attack/Anomaly Detection

Johnson [9] proposes a practical work based on graph analytics adopting a new metric that measures the vulnerability of a network environment with respect to the risk of privilege escalation. However, this work focuses on the proposal of a single graph-based metric, that is intended only as a means to evaluate the vulnerability of a network to cyber-attacks but does not help in detection of cyber-attacks.

Friedberg [10] proposed an anomaly detection system for identifying cyber-attacks from several security logs. Their approach requires a huge number of logs that are often impractical to obtain, and the output of their proposal may be extremely difficult to interpret for a security analyst, since their approach is agnostic to the given input. On the other hand, our focus on network traffic is more practical, as it detected suspicious network activities possibly related to key phases of cyber-attacks.

There are also many examples of cyber-attacks detection systems based on graphs [11], and several attempts were made to use the NetFlow protocol to detect DoS attacks such as Smurf, and worms such as W32.Blaster Worm and Red Worm. However, none of them are able to leverage the modeling of Netflow based data using property-graphs.

B. Graph Algorithms in Deep Feature Representation

Recently, researchers have started to apply deep learning to network structure representation learning. Several proposals have been made to learn a low-dimensional vector representation of individual nodes by considering their neighborhood [12]. Learning a deep graph feature representation lies at the heart and success of many within-network and across-network machine learning tasks such as node and link classification, anomaly detection, link prediction, and many others. Methods capable of learning such representations have many advantages over feature engineering in terms of cost and effort.

The success of graph-based machine learning algorithms depends largely on data representation. Several proposals have been made to learn a low-dimensional vector representation of individual nodes by considering their neighborhood. Deep learning techniques [13] have also improved graph kernels for graph structure learning. These methods focus only on the local structure of a graph and graph kernels require expensive pairwise comparisons. These works focus on unsupervised learning or semi-supervised learning and generating features of different nodes in a graph rather than the embedding of the edges.

VII. Conclusion

We proposed DeepGFL, a general and flexible framework for extracting deep graph features to distinguish network attack flow from benign flow based on the raw features of network flow. Each deep graph feature learned by DeepGFL corresponds to a composition of feature functions applied over the raw flow features. Thus, deep graph features learned by DeepGFL are interpretable and naturally generalize for network representation learning tasks and suitable for network flow graphs. Using this framework, we can detect various of network attack scenarios based on deep graph features.

Acknowledgments

The authors would like to thank Bo Jiang, Mingyi Chen and Baoxu Liu for their helpful comments on this paper and the anonymous referees for their valuable comments and corrections. This research was supported by the Natural Science Foundation of China (No. 61702508, No.61602470, No.61572481).

References

[1] Nasr M, Houmansadr A, Mazumdar A. Compressive Traffic Analysis: A New Paradigm for Scalable Traffic Analysis[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017: 2053-2069.

[2] Nguyen H, Franke K, Petrovic S. Improving effectiveness of intrusion detection by correlation feature selection[C]//Availability, Reliability, and Security, 2010. ARES'10 International Conference on. IEEE, 2010: 17-24.

[3] Rossi R A, Zhou R, Ahmed N K. Deep feature learning for graphs[J]. arXiv preprint arXiv:1704.08829, 2017.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT Press, 2016.

[5] R. A. Rossi, R. Zhou, and N. K. Ahmed, “Estimation of graphlet statistics,” in arXiv preprint, 2017, pp. 1–14.

[6] N. K. Ahmed, T. L. Willke, and R. A. Rossi, “Estimation of local subgraph counts,” in IEEE BigData, 2016, pp. 586–595.

[7] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, “Efficient graphlet counting for large networks,” in ICDM, 2015, p. 10.

[8] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Purtogal, January 2018.

[9] Johnson J R, Hogan E A. A graph analytic metric for mitigating advanced persistent threat[C]//Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on. IEEE, 2013: 129-133.

[10] Friedberg I, Skopik F, Settanni G, et al. Combating advanced persistent threats: From network event correlation to incident detection[J]. Computers & Security, 2015, 48: 35-57.

[11] E. Bou-Harb and M. Scanlon, “Behavioral service graphs: A formal data-driven approach for prompt investigation of enterprise and internet-wide infections,” Digital Investigation, vol. 20, pp. S47–S55, 2017.

[12] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In Proc. of SIGKDD, 2016.

[13] Yanardag P, Vishwanathan S V N. Deep graph kernels[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015: 1365-1374.

[14] Shalev-Shwartz S, Singer Y, Srebro N, et al. Pegasos: Primal estimated sub-gradient solver for svm[J]. Mathematical programming, 2011, 127(1): 3-30.