2019 IEEE Security and Privacy Workshops (SPW)

Deep in the Dark - Deep Learning-based Malware Traffic Detection without Expert Knowledge

Gonzalo Marín*†, Pedro Casas*, Germán Capdehourat†
*AIT Austrian Institute of Technology - Vienna, Austria
†IIE-FING, Universidad de la República - Montevideo, Uruguay

***Abstract*—With the ever-growing occurrence of networking attacks, robust network security systems are essential to prevent and mitigate their harming effects. In recent years, machine learning-based systems have gain popularity for network security applications, usually considering the application of *shallow* models, where a set of expert handcrafted features are needed to pre-process the data before training. The main problem with this approach is that handcrafted features can fail to perform well given different kinds of scenarios and problems. Deep Learning models can solve this kind of issues using their ability to learn feature representations from input raw or basic, non-processed data. In this paper we explore the power of deep learning models on the specific problem of detection and classification of malware network traffic, using different representations for the input data. As a major advantage as compared to the state of the art, we consider *raw* measurements coming directly from the stream of monitored bytes as the input to the proposed models, and evaluate different raw-traffic feature representations, including packet and flow-level ones. Our results suggest that deep learning models can better capture the underlying statistics of malicious traffic as compared to classical, shallow-like models, even while operating *in the dark*, i.e., without any sort of expert handcrafted inputs.**

***Index Terms*—Deep Learning; Network Traffic; Raw Measurements; Malware Detection.**

I. Introduction

The popularity of Deep Learning (DL) models has seen tremendous growth because of their power to achieve great results in many signal processing problems, such as computer vision, audio processing, natural language processing, etc. One of the key reasons making DL models widely used today is the increasing computational power and the availability of larger datasets to perform the training.

The dramatic impact and massive breakthrough of DL can be associated with 2012's ImageNet large scale visual recognition challenge (ILSVRC2012), in which Krizhevsky et al. [2] presented for the first time a model based on a deep convolutional neural network (CNN) for this task, winning the challenge by a wide margin (lowering the state-of-the-art error rate from 26.1% to 15.3%). One of the most powerful characteristics of DL models is their ability to learn feature representations from input raw or basic, non-processed data. For example, a CNN trained for image classification can learn to recognize edges and more complex structures along the sequence of neural layers, using as input only the image RGB pixel values (refer to [1] for a detailed example).

Despite the success of DL models, shallow machine learning models are usually applied when it comes to the analysis of network traffic measurements. When using these models, a feature vector of expert-handcrafted features is usually built in order to achieve the best results. This is actually the critical step on which the success of the model depends. There are different problems when addressing network traffic measurements tasks using shallow machine learning approaches. First, the lack of a consensual labeled *raw* traffic, full-packet capture dataset to train these models (e.g. due to privacy policies in the data); second, the lack of a consensual set of input features to tackle specific targets, such as network security, anomaly detection, traffic classification, etc.; third, the continuous changing in network measurements statistics that may cause static handcrafted features to fail. To improve these limitations, we explore in this paper the end-to-end applications of DL models to complement traditional approaches for network measurement analysis, using different representations of the input data. We particularly focus on the problem of malware traffic detection and classification through deep neural networks, using raw, bytestream-based data as input. Inspired on previous recent work on this domain [13], we present and evaluate different DL architectures and different input representations showing outstanding performance on the analysis of raw bytestream packet data for network malware traffic detection.

The rest of the paper is organized as follows: in Section II we present a brief state-of-the-art on DL models applied to the analysis of network traffic measurements; in Section III we present the different DL approaches selected and evaluated in this work, for the specific detection of malware traffic; in Section IV we discuss the detection performance achieved by these approaches, comparing them with traditional, shallow-like based models, using domain expert knowledge to craft the input features; in Section V we introduce a variation of the detection problem using a multi-class, malware classification approach, and present concluding remarks in Section VI.

II. State-of-the-art

The application of shallow, machine learning models to general network measurement problems is largely extended in the literature. There are a couple of extensive surveys and papers on network measurement problems such as network anomaly detection [5] [4] - including machine learning-based approaches [3], machine learning for network traffic classification [7] and network security [6].

Some DL approaches have been recently shown good performance, mainly associated to traffic classification tasks. In

DOI 10.1109/SPW.2019.00019

2015, Z. Wang *et al.* [17] presented a Deep Neural Network for feature learning using feed-forward networks and Stacked Auto-Encoders (SAE) to perform network protocol recognition over a dataset made up of TCP flows from an internal network. In 2017 there have been some works over the subject. W. Wang *et al.* presented two models based upon a 2D-CNN [16] and 1D-CNN [15], in which the authors transform network flows and sessions to images to work as an input for the CNN models, using either the information of all the layers, or only from the application layer. Lotfollahi *et al.* [12] presented an approach for encrypted traffic classification using SAE and 1D-CNN at the packet level. M. Lopez-Martin *et al.* [11] presented different DL architectures based on CNN and LSTM networks to perform traffic classification using self-collected network traces. Recently, in 2018, Radford *et al.* [14] presented an anomaly detection model using a LSTM network from network traffic logs for cyber-security. In [13] authors introduced *RawPower*, a DL architecture showing outstanding performance on the analysis of raw bytestream data for network anomaly detection.

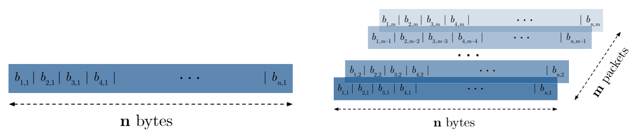
Most of these proposals use DL models after some preprocessing of the data was made, or after some set of handcrafted features was built in order to extract meaningful input for the DL models. The main contribution of our approach is that of using completely expert-knowledge-independent inputs for the prediction and classification tasks - just the raw bytestream, opening the door to a broad set of potential applications of DL for networking problems.

III. Deep Learning for Malware Detection

Our goal is to train a DL model with the stream of incoming bytes without requiring any preprocessing step or domain expert intervention, to make the approach generic and flexible. As we said before, we particularly focus on the problem of malware traffic detection and classification, using raw, bytestream-based data as input. The input representation of the data, as well as the network architecture, are both key facts when building a DL model. Since we want to evaluate the feature representation power of the model from non-processed data, we consider two types of raw representations: packets and flows. In both cases we consider decimal normalized representation of every byte of every packet as a different feature. Since different packets can have different sizes, we set a fixed threshold n to trim each incoming packet to the first n bytes, after removing potentially biasing byte information such as MAC and IP addresses. All packets with size larger than n bytes are trimmed, and packets with smaller size are zero-padded. Next we describe the two raw representations based on packets and flows, along with the corresponding network architectures.

A. Input Representations

In the packet approach, we consider each packet as a different instance, while in the flow approach we consider a group of packets –that make up the flow– as an input for the network, i.e., we build a tensor made of packets that represents a flow as an input instance. Both representations are depicted in Fig. 1. For the *Raw Packet* representation, we have to choose the number of bytes from the packet to consider (n), while in the flow representation we also have to set the number of packets per flow to consider (m). This is because, naturally, different packets and flows can have different sizes. The packet approach is inspired by previous work [13].



$b_{1,1}$ | $b_{2,1}$ | $b_{3,1}$ | $b_{4,1}$ | ... | $b_{n,1}$
$\mathbf{n}$ bytes
$\mathbf{m}$ packets
$\mathbf{n}$ bytes

(a) Packet representation for the input data. The shape of the input data is (N, n): N is the number of instances –packets– and n the number of steps –bytes–.

(b) Flow representation for the input data. A tensor of size (N, m, n) where N represents the number of instances –flows–, m the number of channels –packets– and n the number of steps –bytes–.

Figure 1: Different input representation for the DL models.

Since both malware and normal captures are gathered under controlled conditions, there is some bias in the IP and transport protocol headers that are not representative of *in the wild* traffic. This is the case, for example, of fixed values for IP addresses and ports and even some of the transport protocol flags. For this reason, we take the *payload* of every packet as the key information to analyze and to build the dataset. Afterwards, we set a fixed threshold for the parameter n to trim each incoming packet to the first n bytes of payload. All packets with size larger than n bytes are trimmed, and packets with smaller size are zero-padded at the end. For the number of packets per flow, we fixed a number m and took the first m packets of the flow, discarding the rest.

B. DL Architectures - Raw Packets

The architecture of the DL network for the *Raw Packets* input representation is shown in Fig. 2. It consists of two 1D-CNN layers of 32 and 64 filters of size 5, respectively; a max-pooling layer of size 8; a LSTM layer of 200 units, returning the outputs of each cell ("return sequences" mode on); and finally, two fully-connected (FC) layers of 200 units each. A binary cross-entropy is used as the loss function. Spatial and normal batch normalization layers are added after each 1D-CNN and FC layers to ease the training process. Dropout layers are also used to add regularization to the model.

C. DL Architectures - Raw Flows

When deciding on the network architecture for the *Raw Flows* approach, we note that the number of instances to deal with when operating at the flow level is by far much smaller than in the case of packet-based inputs; as a consequence, the capacity of the model does not have to be as high as in the *Raw Packets* case. The architecture in this case consists of one 1D-CNN layer of 32 filters of size 5 and two fully-connected layers of 50 and 100 units each. Also, binary cross-entropy is used as the loss function. The architecture is shown in Fig. 3.

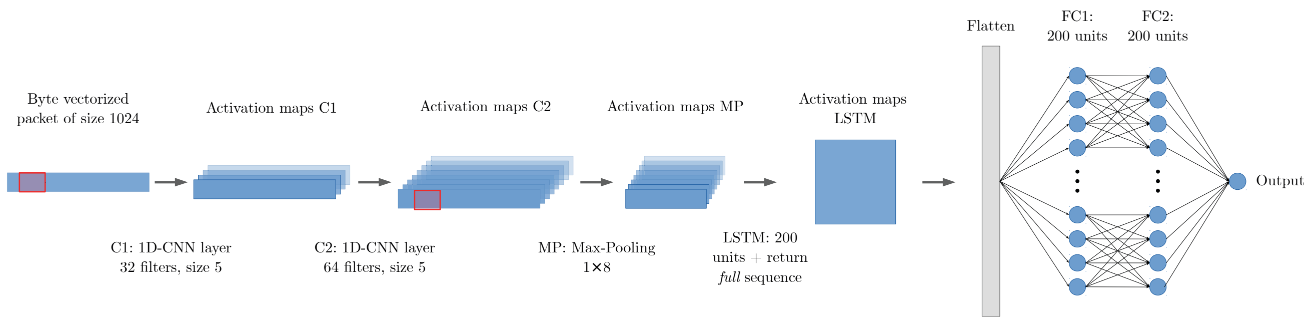

Byte vectorized packet of size 1024
Activation maps C1
Activation maps C2
Activation maps MP
Activation maps LSTM
Flatten
FC1: 200 units
FC2: 200 units
Output
C1: 1D-CNN layer 32 filters, size 5
C2: 1D-CNN layer 64 filters, size 5
MP: Max-Pooling 1×8
LSTM: 200 units + return *full* sequence

Figure 2: DL architecture for *Raw Packets* representation.

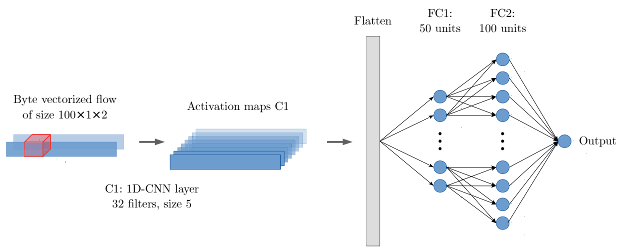

Byte vectorized flow of size 100×1×2
Activation maps C1
Flatten
FC1: 50 units
FC2: 100 units
Output
C1: 1D-CNN layer 32 filters, size 5

Figure 3: DL architecture for *Raw Flows* representation.

Representation	Dataset size	n (bytes)	m (packets)
Raw Packets	248, 850	1024	–
Raw Flows	67, 494	100	2

Table I: Parameters selection for building the input representation for training the DL models.

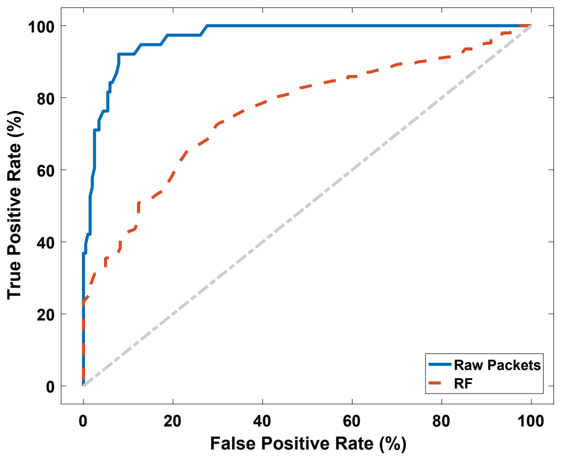


Figure 4: Malware detection performance. Raw packet inputs, DL vs. RF model.

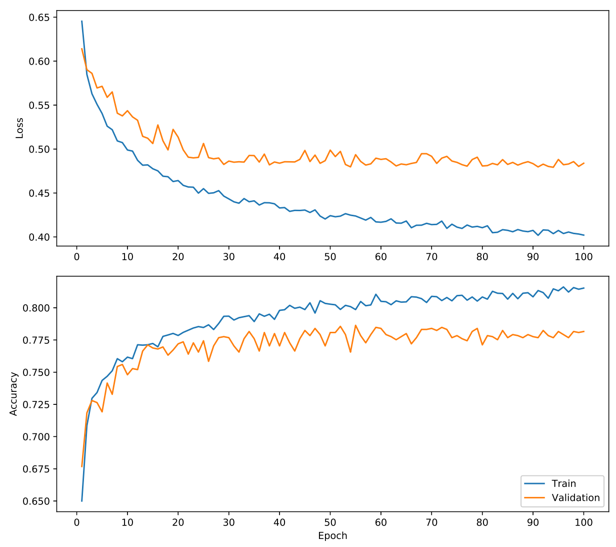
IV. Experimental Evaluations

We evaluate the different proposed DL architectures and input representations using real network measurement datasets, publicly available through the *Stratosphere IPS Project* of the CTU University of Prague in Czech Republic [10]. In this section we focus exclusively on the problem of malware detection, posing it as a binary classification problem: either normal instance or malware. To show the main advantages of the proposed approaches, we pose ourselves three evaluation questions: (i) is it possible to achieve high detection accuracy with low false alarm rates using the raw-input, DL-based models?; (ii) are the proposed DL-based models better than the commonly used shallow models for malware detection, when feeding them all with raw inputs (e.g., bytestreams)?; and (iii) how good are the raw-input, DL-based models as compared to a classical approach for malware detection, where shallow models take as input specific hand-crafted features based on domain expert knowledge?.

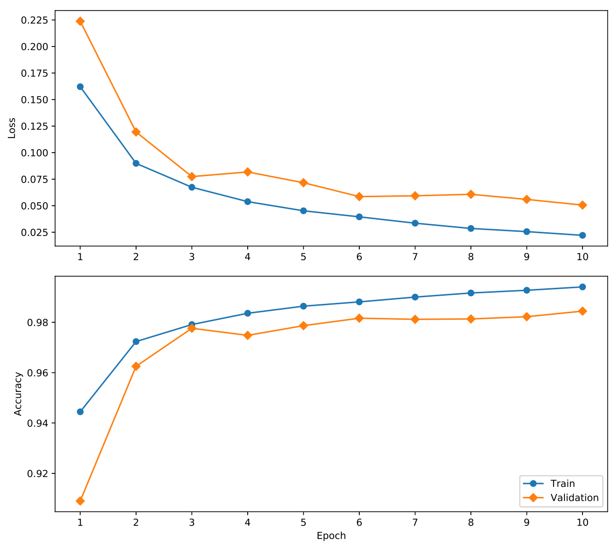
A. DL vs. Shallow Models with Raw Inputs

We begin with a simple evaluation scenario, detecting malware at the packet level. We take a first dataset previously used in the literature [16], referred to as the `USTCTFC2016` dataset. This dataset consists of two groups of labeled `pcap` files from malware and benign applications. The malware files contain 10 types of malware traffic from public websites, collected from a real network environment by researchers of the Czech Technical University in Prague (CTU) [10]. The benign traffic contains 10 types of normal traffic which were collected using Ixia BreakingPoint (https://www.ixiacom.com), a network traffic simulation platform. We take every packet from each labeled `pcap` as part of the dataset, using decimal normalized representation of every byte as a different feature. We set $n = 1024$ bytes; each packet is finally labeled as either benign or malware, i.e., we consider a binary classification problem. The total dataset consists of one million samples (i.e., trimmed packets), half of them benign and half of them coming from the malware traces. We split the dataset on 80% of the samples for training, 10% for validation and 10% for testing purposes. We built the model using the Keras framework running on top of TensorFlow, using the Big-DAMA platform [9], a big-data cluster for analyzing network traffic data with machine learning models.

Fig. 4 presents the initial results obtained by the Raw Packet

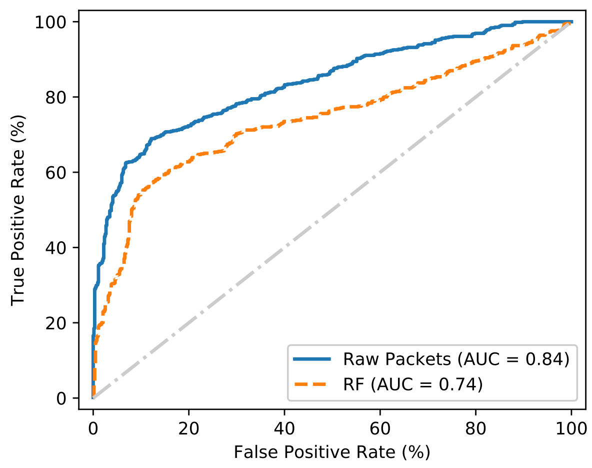


(a) *Raw Packets* representation.

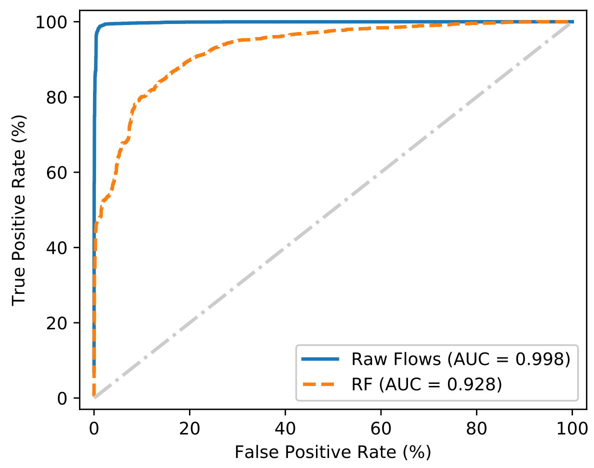


(b) *Raw Flows* representations.

Figure 5: Learning performance (loss and accuracy evolution after each epoch) for *Raw Flows* and *Raw Packets* approaches.



(a) *Raw Packets* representation.



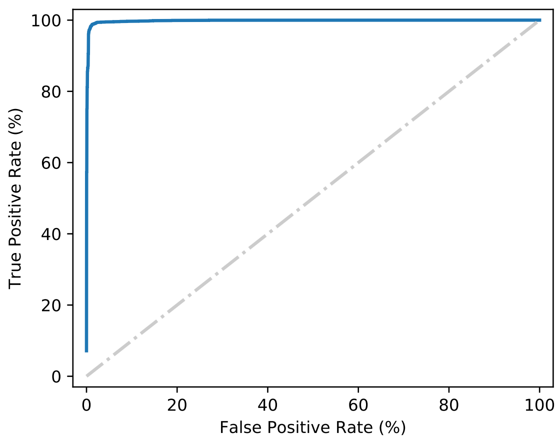
(b) *Raw Flows* representation.

Figure 6: Malware detection performance using *Raw Packets* and *Raw Flows* representations.

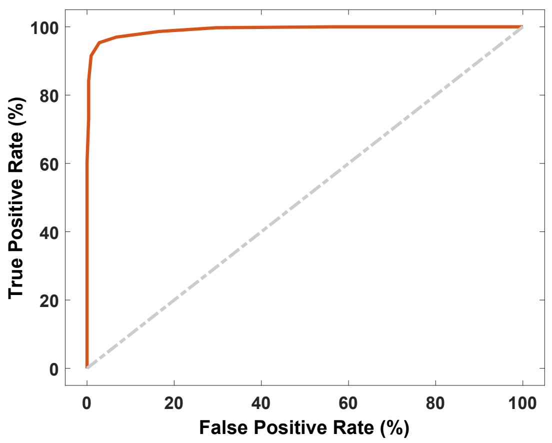
DL model in the detection of malware packets, in the form of a ROC curve. The model is compared to a Random Forest (RF) one, using exactly the same input features and an internal architecture of 100 trees. We choose a RF model based on the generally outstanding detection performance shown by the model in our previous work [8], using domain expert input features. The DL model can detect more than 70% of the malware instances with a false alarm rate below 3%, largely outperforming the RF model. Indeed, when applying the RF model with the same set of raw, non processed input features, the obtained results are very poor. These results are highly encouraging, as they point to the ability of the DL-based model to better capture the underlying statistics of the malware, without requiring any specific handcrafted feature set. Still, the absolute detection performance results are not good enough to rely on such a DL model with raw packet inputs for malware detection in the practice.

B. *Packet vs. Flow Representation Performance*

We take a step further an consider a similar comparison as before, but considering now both raw packet and raw flow representations as input. We take again publicly available datasets from CTU, but to test on different scenarios, we extend the mix of data by considering multiple `pcap` files. We consider the same 10 different types of malware captures as before, adding now 16 types of normal captures, choosing the



(a) *Raw Flows* representation.



(b) Expert-Knowledge based input features.

Figure 7: Malware detection performance using *Raw Flows* vs expert-knowledge-based inputs.

same number of packets for each capture to build a balanced dataset (50% malware packets and 50% normal packets).

In Table I we show the selection of parameters for each input representation. These parameters where selected after conducting a statistical analysis over the dataset. We built two different datasets to fit each one of the considered input representations. The dataset for the *Raw Packet* representation, after removing duplicate instances, consists of roughly $250,000$ instances. For the *Raw Flows* representation, the dataset consists of about $68,000$ instances. Both datasets are split according to the same scheme as before: 80% of the samples for training, 10% for validation and 10% for testing.

The learning processes for both approaches is described in Fig. 5. In both cases we used mini-batches for the parameters update and we trained the models over several epochs (being an epoch a single pass-through over the complete training dataset). In the case of the *Raw Packets* representation, the training was held over 100 epochs, while in the case of *Raw Flows* we used 10 epochs. We used Adam as the optimizer function annealing the learning rate over time. The performance metric chosen in both cases was the accuracy, since the dataset is balanced. For the *Raw Packets* representation we achieved 77.6% of accuracy over the test set; while in the case of *Raw Flows* we achieved an accuracy of 98.6% also over the test set. Note that the learning process performs better when operating at the flow level, as there is some potential overfitting for this scenario using the packet-level representation.

As we did before, Fig. 6 compares the detection performance of both models against a RF model using exactly the same (raw) input features - in the *Raw Flows* case, we flatten the data to fit the input to the RF. In both cases, the internal architecture of the RF consisted of 100 trees using different pruning techniques to prevent over-fitting (e.g., maximum depth, maximum number of instances per leaf, etc.). Once again, we observe a clear out-performance of the DL architectures as compared to the RF models, particularly when operating at the flow level. For *Raw Packets* representations, the detection model can detect about 65% of the malware traffic packets with a false alarm rate below 3%, with an overall out-performance of nearly 15% as compared to the RF. Note that in this evaluation, the differences in terms of performance are not as important as in Fig. 4.

The main differences occur when operating with *Raw Flows* representations, where the DL model can detect as much as 98% of all malware flows with a false alarm rate as low as 0.2%. This suggests that, when operating at the flow level, such raw input representation and associated DL architecture can actually provide highly accurate results, applicable in the practice.

C. *Domain Knowledge vs. Raw Inputs*

The last step of the evaluations tries to answer the third question regarding the goodness and advantages of the proposed approach w.r.t. the standard approach for machine-learning based malware detection. In particular, we study how good is the raw-flows, DL-based model as compared to a RF-based model, the latter using as input specific hand-crafted features based on domain expert knowledge.

The standard approach for detection of malware and network attacks in networking traffic is to rely on flow-level features, using traditional in-flow packet measurements such as traffic throughput, packet sizes, inter-arrival times, frequency of IP addresses and ports, transport protocols and share of specific flags (e.g., SYN packets), etc. We therefore build a set of almost 200 of these features to feed a RF model. Note that besides using traditional features such as min/avg/max values of some of the input measurements, we also consider their empirical distribution, sampling the empirical distribution at many different percentiles. This provides as input much richer information, as the complete distribution is taken into account. We take the same dataset used in Sec. IV-B for training and testing purposes.

Botnet	Protocol	Activity
Neris	IRC	spam, click fraud
Rbot	IRC	DDoS
Virut	HTTP	spam, port scan

Table II: Protocols and attacks performed by different kinds of botnets in the scenarios chosen from the CTU dataset.

Class	Accuracy	Precision	Recall	F_1 score
Normal	0.878	0.621	0.878	0.727
Neris	0.635	0.814	0.635	0.714
Rbot	0.999	1.000	0.999	1.000
Virut	0.547	0.679	0.547	0.606

Table III: Performance metrics for the DL model for the malware classification problem.

Fig. 7 reports the results obtained on this scenario. Not surprisingly, the RF using expert domain features achieves highly accurate detection performance, detecting about 97% of all the malware instances with less than 1% of false alarms. However, also in this scenario, the DL-based model, using raw flow representations as input, slightly outperforms this domain expert knowledge based detector. As such, we can conclude that DL model can perform as good as a more traditional shallow-model based detector for detection of malware flows, without requiring any sort of expert handcrafted inputs. This of course, shows the great contribution of our approach.

Based on the three sets of evaluations, and recalling that the RF model serves as performance benchmark - based on our previous results showing their performance on malware and network attacks detection tasks [8], we can conclude that the proposed DL model, in particular that using raw flow representations as input, can: (i) provide highly accurate and applicable-in-the-practice malware detection results, (ii) capture the underlying malware and normal traffic models better than a shallow-like, RF-based model, and (iii) provide results as good as those obtained through a domain expert knowledge-based detector, without requiring any sort of handcrafted features.

V. From Malware Detection to Classification

To complement previous malware detection results, in this section we present a variation of the binary classification problem, considering now different sorts of malware attacks traffic as different classes, together with a "normal" class representing benign traffic. In the CTU dataset, each capture represents a different *scenario*, in which different sorts of malware were executed using several protocols (e.g., IRC, HTTP, P2P, etc.) and executing different types of attacks (e.g., DDoS, port scan, click fraud, spam, etc.).

To build the dataset we considered three different malware traffic classes, corresponding to three different types of botnets, named Neris, Rbot, and Virut. Thus, our multiclass classification problem has four different classes: three that represents malware attacks and one that represents normal activity. The activity and protocols used for the scenarios chosen for building our dataset are depicted in Table II. The dataset consists of $160,000$ samples, built in a stratified way, i.e., the dataset is balanced with $40,000$ samples per class.

The DL model used for this task is quite similar to the one used for the *Raw Packets* representation. The difference is that now the activation function used in the last fully connected layer is a Softmax function (a generalization of the binary logistic regression classifier for multiple classes), instead of the sigmoid used for the binary classification. The resulting architecture is depicted in Fig. 8. The corresponding loss function is also different for the multi-class classification problem. In this case we used a categorical cross-entropy, and the learning process was held over 50 epochs. The evolution of the learning process, including loss and accuracy after each epoch, is very similar to the previous raw packets-based model (cf. Fig. 5a), as both operate using a similar input representation.

For this problem, we also compared the performance of the DL model against a Random Forest one, using the same input features for both. In Fig. 9 we show the normalized confusion matrices for both models (values are shown in percentages); the DL model outperforms the RF for all classes. As a sanity check, note that the accuracy of the multi-class problem considering a binary approach (malware vs. normal) holds similar results as the *Raw Packets* approach presented in Sec. IV-B (77.6% vs. 76.5%). To complement, in Table III we show additional performance metrics for the DL model, including accuracy (AC), precision (PR), recall (RC) and F_1 score for each class. Values are computed in a *one-vs-all* schema, in which each class is evaluated against the rest, as if it was a binary problem. It is interesting to note that Rbot botnet is detected with an accuracy of 99.9% while Neris and Virut achieve 63.5% and 54.7% each. It is likely that the fact that both Neris and Virut share spam as an activity attack, could be the root cause behind the bad performance to distinguish one from each other - see Table II. Interesting is the fact that once again, the DL architecture using raw inputs clearly outperforms the RF model, for all variations of malware. As a general conclusion of the multi-class problem, we can observe that the raw packets representation does not provide discriminative enough input for the proposed DL architecture to realize properly classification results, but it still outperforms the benchmark RF model. As part of our ongoing efforts, we are working on both testing the multi-class problem using raw flows representations, as well as on improving classification performance for the raw packets input.

VI. Conclusions

In this paper we have presented two different approaches using Deep Learning for the detection of malware network

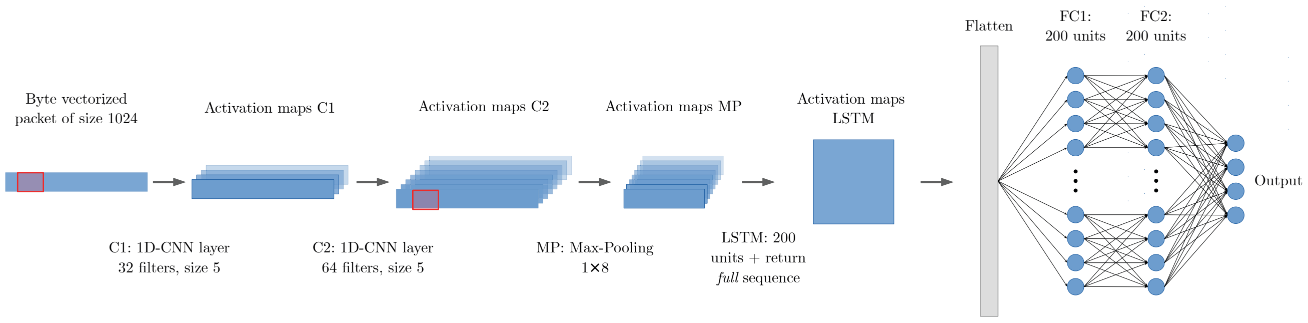
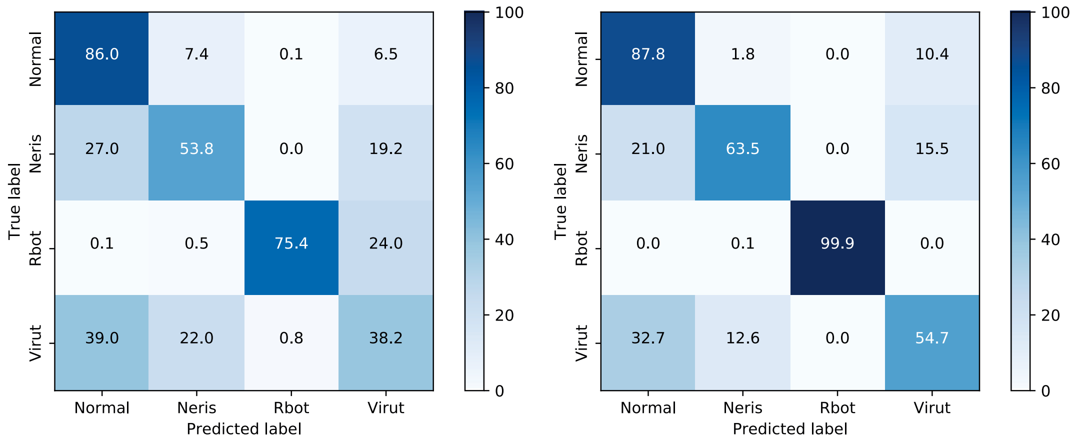

Byte vectorized packet of size 1024
Activation maps C1
Activation maps C2
Activation maps MP
Activation maps LSTM
Flatten
FC1: 200 units
FC2: 200 units
Output
C1: 1D-CNN layer 32 filters, size 5
C2: 1D-CNN layer 64 filters, size 5
MP: Max-Pooling 1×8
LSTM: 200 units + return full sequence

Figure 8: DL architecture for multi-class classification of malware traffic, using *Raw Packets* representation.



(a) Random Forest.

True \ Predicted	Normal	Neris	Rbot	Virut
Normal	86.0	7.4	0.1	6.5
Neris	27.0	53.8	0.0	19.2
Rbot	0.1	0.5	75.4	24.0
Virut	39.0	22.0	0.8	38.2

(b) DL Multi-Class.

True \ Predicted	Normal	Neris	Rbot	Virut
Normal	87.8	1.8	0.0	10.4
Neris	21.0	63.5	0.0	15.5
Rbot	0.0	0.1	99.9	0.0
Virut	32.7	12.6	0.0	54.7

Figure 9: Normalized confusion matrices (showing percentage values) for both Random Forest and DL models.

traffic, considering *raw* representations of the input network data. Different from traditional, shallow-based approaches, our models operate with raw, byte stream inputs, without requiring any type of handcrafted, expert domain knowledge-based input features or feature engineering, providing an extremely powerful approach. Evaluations show that using *Raw Flows* as input to the DL models results in much better performance than using *Raw Packets*, achieving detection performance which is comparable - or even better, than the obtained by expert-domain knowledge. We also presented a variation of the binary classification model using a multi-class approach to discriminate between different types of malware. In all cases, DL models outperform a strong RF model used as benchmark, using exactly the same raw input features. This demonstrates the power of the DL models to better capture the underlying statistics of malicious traffic, as compared to more classical, shallow-like models.

References

[1] Simonyan, K., Vedaldi, A., and Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR abs/1312.6034* (2013).

[2] Krizhevsky, A. et al., Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. NIPS*, 2012.

[3] Ahmed, T., Oreshkin, B., Coates, M. Machine Learning Approaches to Network Anomaly Detection. In *Proc. USENIX SYSML*, 2007.

[4] Bhuyan, M. H., et al. Network Anomaly Detection: Methods, Systems and Tools. In *IEEE Comm. Sur. & Tut.*, vol. 16 (1), pp. 303–336, 2014.

[5] Mohiuddin, A., Naser, A., Hu, J. A Survey of Network Anomaly Detection Techniques. In *J. of Net. and Comp. App.*, vol. 60, pp. 19–31, 2016.

[6] Buczak, A. L., Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. In *IEEE Communications Surveys & Tutorials*, vol. 18 (2), pp. 1153–1176, 2008.

[7] Nguyen, T. T., Armitage, G. A Survey of Techniques for Internet Traffic Classification using Machine Learning. In *IEEE Communications Surveys & Tutorials*, vol. 10 (4), pp. 56-76, 2008.

[8] Casas, P., D'Alconzo, A., Settanni, G., Fiadino, P., Skopik, F. (Semi)-supervised machine learning approaches for network security in high-dimensional network data. In *Proc, ACM CCS*, 2016.

[9] Casas, P., Soro, F., Vanerio, J., Settanni, G., D'Alconzo, A. Network Security and Anomaly Detection with Big-DAMA, a Big Data Analytics Framework. In *Proc. IEEE CloudNET*, 2017.

[10] García, S., et al. An Empirical Comparison of Botnet Detection Methods. *Comput. Secur. 45* (Sept. 2014), 100–123.

[11] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., and Lloret, J. Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access 5* (2017), 18042–18050.

[12] Lotfollahi, M., Zade, R. S. H., Siavoshani, M. J., and Saberian, M. Deep Packet: A Novel Approach for Encrypted Traffic Classification using Deep Learning. *CoRR abs/1709.02656* (2017).

[13] Marín, G., Casas, P., and Capdehourat, G. Rawpower: Deep Learning based Anomaly Detection from Raw Network Traffic Measurements. In *Proc. ACM SIGCOMM SRC*, poster, 2018.

[14] Radford, B. J., et al. Network Traffic Anomaly Detection using Recurrent Neural Networks. *CoRR abs/1803.10769* (2018).

[15] Wang, W., et al. End-to-end Encrypted Traffic Classification with one-dimensional Convolution Neural Networks. In *Proc. IEEE ISI*, 2017.

[16] Wang, W., et al. Malware Traffic Classification using Convolutional Neural Network for Representation Learning. In *Proc. ICOIN*, 2017.

[17] Wang, Z. The Applications of Deep learning on Traffic Identification. In *Black Hat USA, Las Vegas* (2015).