Efficient Learning of Communication Profiles from IP Flow Records

Hammerschmidt, Christian; Marchal, Samuel; State, Radu; Pellegrino, Nino; Verwer, Sicco

DOI
10.1109/LCN.2016.92

Publication date
2016

Document Version
Accepted author manuscript

Published in
Proceedings - 2016 IEEE 41st Conference on Local Computer Networks, LCN 2016

Citation (APA)
Hammerschmidt, C., Marchal, S., State, R., Pellegrino, N., & Verwer, S. (2016). Efficient Learning of Communication Profiles from IP Flow Records. In P. Kellenberger (Ed.), Proceedings - 2016 IEEE 41st Conference on Local Computer Networks, LCN 2016 (pp. 1-4). Los Alamitos, CA: IEEE. https://doi.org/10.1109/LCN.2016.92

Efficient Learning of Communication Profiles from IP Flow Records

Christian Hammerschmidt*, Samuel Marchal†, Radu State*, Gaetano Pellegrino‡, and Sicco Verwer‡
*SNT, University of Luxembourg
†Aalto University
‡TU Delft
Email: {christian.hammerschmidt,radu.state}@uni.lu; samuel.marchal@aalto.fi; {g.pellegrino,s.e.verwer}@tudelft.nl

***Abstract*—The task of network traffic monitoring has evolved drastically with the ever-increasing amount of data flowing in large scale networks. The automated analysis of this tremendous source of information often comes with using simpler models on aggregated data (e.g. IP flow records) due to time and space constraints. A step towards utilizing IP flow records more effectively are stream learning techniques. We propose a method to collect a limited yet relevant amount of data in order to learn a class of complex models, finite state machines, in real-time. These machines are used as communication profiles to fingerprint, identify or classify hosts and services and offer high detection rates while requiring less training data and thus being faster to compute than simple models.**

I. Introduction

Due to the high volume of data exchanged in modern networks, in-depth analysis of the whole traffic is no longer realistic. A more common approach is to analyze aggregated communication information of which IP flow records is an example. The main challenge lies in the extraction of relevant information from this meta data. In this paper, we focus on the problem of creating a model to classify hosts based on their traffic summary statistics. We refer to this task as behavioral *communication profiling*. Current methods addressing this task use batch processing techniques over large amount of data [1], [2]. This has two drawbacks being the delay induced in model learning due to long period of data collection and the limited complexity of the analysis methods [3] due to space and computation limitation. Consequently, these simple methods are not able to model accurately communication profiles.

To address these limitations, we propose to use complex models for modeling fine grained communication profile with finite state machines. In contrast with previous work [4], [5], we use finite state machines with a stream learning component allowing us to start learning a communication profile in real-time as network traffic is observed. We show that the amount of training data required to learn an accurate communication profile can be determined on the fly, limiting thus data collection time and amount of data to process. We assess that profiles learned from limited IP flow data are as efficient as ones using more training data for the use case of botnet hosts detection. To summarize our contributions:

- We introduce a feature engineering method to aggregate IP flow records into a state space representation, which can be input to a finite state machine (Section III-B);
- We present methods to evaluate the amount of information contained in the training set, which allows to control data collection and selection (Section III-C);
- We validate our techniques on real-world traffic obtaining competitive detection rates (Sections IV and V).

II. Background

A. IP Flow Analysis

IP flows records are statistics from packets exchanged between two hosts. The statistics are collected and aggregated by a specialized device (e.g. a router). We refer to [6] for an overview of the basics of IP flow record data collection. IP flow records are tuples of features including source IP address, source port, destination IP address and destination port to describe the participants. The *start time* and *duration* specify when the flow occurred, and *transport protocol*, *packet counts* and *amount of data* exchanged in both directions summarize the exchange itself. Table I provides a summary of the considered features.

B. Probabilistic Deterministic Finite Automata (PDFA)

Finite state automata are a type of automaton model often used to describe computation and processes in a formal way. We use finite state automata with probabilities, called probabilistic deterministic finite automata (PDFA). Introductions to the field of automaton theory can be found in [7]. A *Probabilistic Deterministic Finite Automaton (PDFA)* is quintuple $A = \langle Q, T, \Sigma, q_0, P \rangle$ where Q is a finite set of states, $T : (Q, \Sigma) \to Q$ are labeled transitions with labels drawn from an *alphabet* Σ, $q_0 \in Q$ is the start state. The probability matrix P gives the probability of observing event $a \in \Sigma$ in state q by $p_{a,q}$. A PDFA starts in the start state q_0 and generates strings by traversing transitions and drawing events using P. For example, the probability of generating abc is given by $p_{a,q_0} p_{b,q_1} p_{c,q_2}$ where $q_1 = T(q_0, a)$ and $q_2 = T(q_1, b)$.

C. State-Merging Algorithms

The task of inferring PDFAs from a given set of observations is to find a PDFA accepting the words representing the observed behavior. Currently, state-merging algorithms are state-of-the-art in learning automatons [8]. Given a set S_+ of observed behaviors encoded as words over an alphabet Σ called the *input sample*, the goal is to find a (non-unique) *smallest* PDFA A that is *consistent* with S_+. A PDFA is considered consistent with S_+ if it satisfies a type of Markov

TABLE I: Features of IP flow records. $Time$ is used to aggregate sliding windows.

Features	Description	Values
$protocol$	transport protocol of the flow	categorical: tcp, udp, etc.
$time$	time since previous flow started	timestamp
$duration$	duration of the flow	time in ms
$pakets$	Count of packets exchanged	numerical
$data_{exc}$	Amount of data exchanged	numerical, in KB
$data_{rec}$	Amount of data received	numerical, in KB

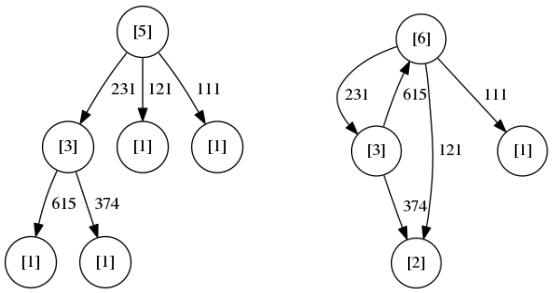


Fig. 1: Left: a prefix tree for a dataset containing the words $\{121, 111, 231, 231.615, 231.374\}$. States contain occurrence counters. Transitions are labeled with the symbol firing them. Right: an automaton obtained by merging the transitions 615 with the root and 374 with the state lead to by 121.

property i.e. for every prefix s from S_+ that reaches the same state q in A, the sample probabilities of future suffixes $P(s' \mid s) = count(ss')/count(s)$ of the states are not significantly different. The size of a PDFA is measured by its number of states.

The starting point for state merging algorithms is the construction of a tree-shaped PDFA A from the input sample S_+. This is called augmented prefix tree acceptor (APTA). Figure 1 (left) shows a prefix tree for a small input sample. It contains all samples from S_+ in a directed graph, using the symbols of the samples in S_+ as labels for the edges. Two samples from S_+ share a path if they share a prefix. The state merging algorithm reduces the size of the automaton iteratively by reducing the tree through merging a pair of states in A, using a heuristic to decide which pairs are best to merge. The merges reduce the size of the automaton (number of states), and introduces loops. Figure 1 (right) depicts the automaton after a state-merging operation.

III. Building Communication Profiles

A. Communication Profiles

A *communication profile* provides a concise description of a participant or a group of participants in a network. We build profiles only using connection-level communication information provided by IP flow records. The main task is to extract the key behavior from the records, and reduce the data into a compact description. Given IP flow records from an unknown source, we can classify; given a known source, we can predict future behavior. Mathematically, a communication profile is a PDFA learned from IP flow records as described in Section II. To infer information about a single host from its IP flow records, we aggregate consecutive flows within a short time period into a single word and use a sliding window technique to obtain sequences of words describing consecutive flows. These words are descriptions of short-term behavior.

B. Encoding IP Records for PDFAs

We obtain input words for PDFAs from IP flow records by converting each IP flow record into discrete symbol and using a sliding window to form a sequence. Each numeric feature of a record, as given in Table I, is put into a discrete bin and represented by the bin number. We calculate percentiles as bin boundaries. E.g. using 25-percentile ranks, we create 4 bins (labelled $[0, 1, 2, 3]$) and calculate feature values such that 25%, 50%, 75% and 100% of the data fall below. For categorical values (*protocol*), we assign each feature value a unique number. The symbolic representation of an IP flow record is the concatenation of the values for all its five features (excluding *time*) and represents a letter e.g. *02213*. After encoding IP flow records as symbols, we aggregate all flows starting within a short, fixed time by sliding a window over all flows, incrementing the start of the window one flow at a time. An input word for a PDFA used as communication profile then consists of a sequence of symbols from a window, where each flow starting within the window's time is represented by a letter.

C. Data Estimation Criteria

The prefix tree (APTA) is the starting point for all state merging learning algorithms. It is a compact way to represent all the training data and offers ideal access to analyze the impact of varying training set sizes on the learning process. The key in minimizing the data needed to learn a model is understanding the error introduced by using a partial sample of the data: It enables us to analyse the quality provided by a partial view of the data with respect to the complete data. We apply two criteria to judge the completeness of the partial sample: For a formal approach, we check the Hoeffding bound (1), a type of concentration inequality [9]. For an informal, application-driven approach we observe the growth in states and transitions when adding more data to the prefix tree, we define this criteria as the *freshness* (2). Equation (1) states the Hoeffding inequality. It bounds the difference between the true mean r of a random variable with the range of the set R with its estimation $\bar{r}$ calculated on a finite sample with low error δ: With probability $1 - \delta$, the error in the estimation $\bar{r}$ only deviates by an ϵ from r. The *true mean* r is the mean calculated on all, possibility infinite samples.

$$r \leq \bar{r} - \epsilon \text{ with prob } 1 - \delta \text{ where } \epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \quad (1)$$

We chose the one-sided upper bound, as it would be most helpful to reason about decisions heuristics applied in state-merging algorithms take. The estimation is sub-linear in terms of the confidence δ and quadratic in sample number for precision ϵ. We apply this technique to the APTA by estimating the relative frequency $\frac{c_i}{n_s}$ of transition i in each state s where $n_s = \sum_{c_i \in s} c_i$. This allows us to bound the error in

TABLE II: Scenarios (ID) composition summary. Records are labeled as background, malicious (bnet) or normal traffic.

ID	#Flows / Duration / Size	Malware (#bots)	Class Distribution back / bnet / norm
10	1,309,791 / 4.75 hrs / 73GB	Rbot (10)	90.7 / 8.1 / 1.2
11	107,251 / 0.26 hrs / 5.2GB	Rbot (3)	89.9 / 7.6 / 2.5
12	325,471 / 1.21 hrs / 8.3GB	NSIS.ay (3)	97 / 2.3 / 0.7

the empirical probability distribution defined by occurrence counts.

IV. Experiments

Our experiments are designed to determine whether a full data representation can be obtained from a partial view of the data by observing *freshness* and the Hoeffding bound to judge prefix tree completeness. Afterwards, we empirically validate this dataset reduction method by learning communication profiles from the obtained sets. We compare their performance in host classification with profiles trained on full training sets.

A. Dataset and Data Preparation

We use a publicly available dataset of manually labeled IP flow traces [10]. It contains real communications from hosts running botnet malware as well as background and legitimate traffic and is organized in several scenarios (ID), each running one or more infected hosts connected to the Internet. We chose scenarios (Table II) that run multiple infected hosts at the same time, allowing us to repeat the same analysis on different instances of the bots. The scenarios differ in characteristics: due to spamming and flooding, some scenarios contain many flows despite few hosts, whereas in others much less traffic per host is captured. The background traffic is real legitimate traffic from other participants in the network.

The IP flow records (Table II) are encoded using the features stated in Table I. Numeric attributes are discretized by assigning a number according to the percentile its value is in. The percentiles themselves are obtained by selecting a random subset of IP addresses from normal traffic (norm) to calculate the statistics. Any knowledge transfer is prevented by excluding these IP addresses from any further experiments. All flows irrespective of their duration, starting within $t = \tau\ ms$ are collected in a window to obtain short term interaction patterns of each IP address. We advance the window on a per-flow level. The duration τ is chosen using the streaming data analysis. This process can be done in real-time as the completed flows are exported.

B. Streaming Data Collection

We observe two different criteria for stopping data collection: In an application-driven approach, we observe the *freshness* Δ of samples w_s with respect to an APTA A. We define it as the ratio $\frac{[\![w]\!]}{|A|}$ of number $[\![w]\!]$ of states newly created in APTA A when adding sample w versus the total number of states $|A|$ in APTA A. Here, $[\![\cdot]\!]$ denotes the length of the word w minus the length of its longest prefix in A. When w is a set, we define $[\![w]\!] = \sum_{w_i \in s} [\![w_i]\!]$ as the sum of states created from the samples in the set. Adding samples that are already contained or have large prefixes in the tree only adds little extra information. The freshness ranges between 0 and 1, and low values indicate that the sample already has many duplicates, or at least long prefixes in the APTA. It serves as an indicator: if it falls below a threshold, the prefix tree already contains most of the data. Because this measure does not guarantee good estimates of the transition probability in each state, we also use a statistics-driven approach: empirical distributions in the states of the APTA have to be bounded by the Hoeffding bound with varying thresholds. The more states have distributions bounded, the better the APTA summarizes the true source.

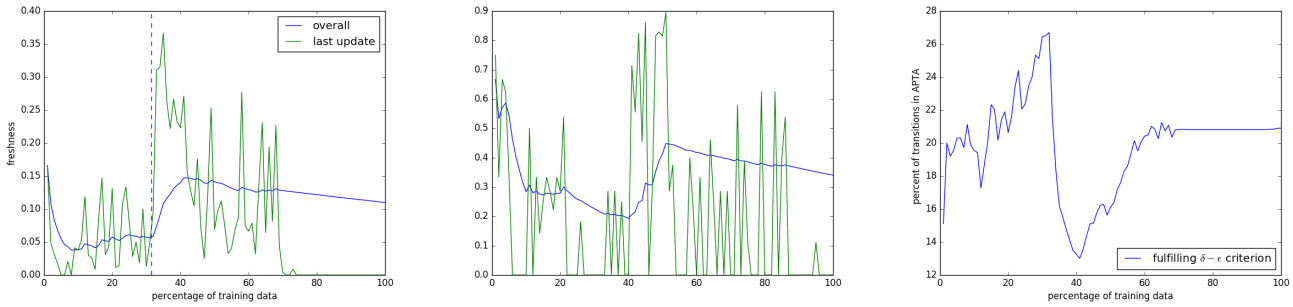
C. Profiling Behavior

We learn communication profiles with the `dfasat` software package [11] using Alergia and Overlap heuristics. The goal is to obtain a small automaton that can reliably distinguish legitimate from botnet sources. The classification task focuses on hosts, not individual traffic flows. We use the full training sets, as well as smaller training sets obtained from an analysis of freshness and Hoeffding bounds on local distributions to learn communication profiles. To judge whether a host is malicious or not, we evaluate its associated communication profile, an APTA A, by calculating its acceptance rate: the ratio of accepted versus rejected windows from an evaluation set. A preliminary analysis showed that an acceptance ratio exceeding 75% any time after the first 25 windows is a good threshold to classify hosts as malicious.

V. Results

A. Streaming Data Collection

We chose a small alphabet size obtained through few bins (4 per feature) and short windows ($\tau = 20\ ms$). An interesting observation across the different scenarios is the non-monotonicity of freshness. It clearly illustrates that the global behavior of a host is composed of several small, different behaviors. This property is captured by PDFAs, which can have multiple loops with transitions of high probability, connected by transitions of lower probability. This is particularly easy to see in Figure 2(a), indicated by a vertical dashed line: after adding increasingly less new information to the prefix tree, the updates at the 32% mark of the training set add a new behavior. The increase in freshness shows that words inserted encode behavior without prefixes in the APTA, i.e. previously unseen behavior. This is also visible in a plot of the states inserted into the prefix tree, i.e. the length of the samples, and indicates that windows start to contain more words. The dataset description of Scenario 10 lists a sequence of bandwidth increases and a switch from a UDP-based flood attack to an ICMP-based attack. The former did not use up the full bandwidth, the latter did. This makes extreme values and monotonicity of freshness an interesting candidate for clustering behavior. Figure 2(c) shows the fraction of transitions fulfilling the Hoeffding bounds for a weak choice of parameters, $\delta = 15\%$ and $\epsilon = 0.15$. In neither scenario



(a) (b) (c)

Fig. 2: Overall freshness in Scenario 10 (a) and Scenario 12 (b). The blue line shows the development of the overall freshness, the green line depicts the freshness of the last update adding the next 1% of the training data to the APTA. The dashed vertical line indicates a point of change: local updates suddenly contain a lot of new samples without prefixes, or much longer samples. The Hoeffding inequality applied on transitions in the APTA, using $\delta = 15\%$ and $\epsilon = 0.15$, depicted for Scenario 10 (c).

TABLE III: Results summarized. The environment contains 48 benign hosts in total. We trained on 1 of the 10, respectively 3, hosts in the dataset and detect the others.

Experimemt	Alergia TP / FP / Pr	Overlap TP / FP / Pr
Baseline in Scenario 10	6 / 0 / 1	7 / 0 / 1
Baseline in Scenario 11	2 / 0 / 1	2 / 0 / 1
Baseline in Scenario 12	1 / 0 / 1	1 / 0 / 1
48% in Scenario 10	6 / 0 / 1	7 / 0 / 1
12% in Scenario 11	0 / 0 / 0	0 / 0 / 0
50% in Scenario 11	2 / 0 / 1	2 / 0 / 1
52% in Scenario 12	1 / 0 / 1	1 / 0 / 1

the ratio of transition bounded correctly exceeded 30%. As a distribution-free bound, is conservative for our use-case.

B. Profiling Behavior

We use the training datasets determined in the previous step to learn PDFAs as communication profiles. Communication profiles trained on all IP flow records of one malicious IP address in each scenario are the baseline. By inspecting the freshness, we chose 48% of Scenario 10, and 52% of Scenario 12 training data. For both cases, Figure 2(a) and 2(b) show a plateau in global freshness, and the freshness of local updates is also low. In Scenario 11, freshness keeps increasing until the end, but is very low ($\Delta < 0.13$). We chose two splitting points: the low point of freshness at 12% of the training data ($\Delta = 0.03$), and for the lack of another extreme point, we also split at 50%. Table III summarizes the results: true and false positives (TP/FP) and precision (Pr), a ratio of $\frac{TP}{TP+FP}$ describing how many of the identified hosts were relevant. For all but the 12% split, results for the communication profile learned from the reduced set are the same as from the baseline. It is very likely that the learning algorithm can infer the core structure from the reduced set and generalize enough. The inability to detect the malicious hosts in Scenario 11 with only 12% of the training data is not surprising. Just observing the freshness can be deceptive: a highly redundant representation of additional data can add valuable data to discriminate hosts, but does so at slow rate.

VI. Discussion and Conclusion

Overall, manually inspecting plots of freshness to decide on smaller training datasets yields good results. We think that it can serve as a tool during preprocessing, just as the Elbow method which is used by manual inspection for clustering. We are currently working on an algorithmic solution to automatically identify the splitting point, as well as using Kullback-Leibler divergence measures to detect a wider range of change-points, such as slow drifts in the occurrence of existing behaviors.

References

[1] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Internet traffic behavior profiling for network security monitoring," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1241–1252, 2008.

[2] M. Jaber, R. Cascella, and C. Barakat, "Using host profiling to refine statistical application identification," in *Proceeding of IEEE INFOCOM*, 2012, pp. 2746–2750.

[3] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *Proceedings of the IEEE International Congress on Big Data*, 2014, pp. 56–63.

[4] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," in *IEEE S&P*, 2009, pp. 110–125.

[5] T. Krueger, H. Gascon, N. Krämer, and K. Rieck, "Learning stateful models for network honeypots," in *ACM AISEC*, 2012, pp. 37–48.

[6] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, 2014.

[7] J. E. Hopcroft and J. D. Ullman, *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman, 1990.

[8] S. Verwer, R. Eyraud, and C. De La Higuera, "PAutomaC: a probabilistic automata and hidden Markov models learning competition," *Machine learning*, vol. 96, no. 1-2, pp. 129–154, 2014.

[9] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University, 2013.

[10] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, Sep. 2014.

[11] N. Walkinshaw, K. Bogdanov, C. Damas, B. Lambeau, and P. Dupont, "A framework for the competitive evaluation of model inference techniques," in *Proceedings of the First International Workshop on Model Inference In Testing*. ACM, 2010, pp. 1–9.