Computer Networks 129 (2017) 64–78

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Rethinking robust and accurate application protocol identification☆

YiPeng Wang[a,b], Xiaochun Yun[a], Yongzheng Zhang[a,b,*], Liwei Chen[a], Tianning Zang[a,b,*]

[a] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Article history:
Received 24 February 2017
Revised 1 September 2017
Accepted 11 September 2017
Available online 14 September 2017

Keywords:
Application protocol identification
Protocol language model
Networking and network security

ABSTRACT

Protocol traffic analysis is a fundamental problem regarding a variety of networking and security applications, such as intrusion detection and prevention systems, network management systems, and protocol specification parsers. In this paper, we propose ProHacker, a nonparametric approach that extracts robust and accurate protocol keywords from the byte sequences generated by an application protocol, and effectively identifies the protocol trace from mixed Internet traffic. ProHacker is based on the key insight that the *n*-grams of protocol traces have highly predictable statistical nature that can be effectively captured by statistical language models and be leveraged for robust and accurate protocol identification. In ProHacker, we first extract protocol keywords using a nonparametric Bayesian statistical model, and then use the corresponding protocol keywords to classify protocol traces by a semi-supervised learning algorithm. We implement and evaluate ProHacker on real-world traces, and our experimental results show that ProHacker can accurately identify the protocol trace with an average precision of about 99.4% and an average recall of about 99.28%. We compare the results of ProHacker to one state-of-the-art approach, ProWord, and one our previous work, Securitas, using backbone traffic. We note that ProHacker provides significant improvements on precision and recall for online protocol identification.

1. Introduction

1.1. Motivation and problem statement

This paper concerns the network-based application protocol identification based on the packet payload of Internet traffic. Protocol traffic analysis requires the identification of protocol keywords, which are referred to by us as a set of byte subsequences (such as a set of *n*-grams) or their statistical quantities that can be used to distinguish the protocol trace of individual protocols. The protocol keyword forms a fundamental building block of deep packet inspection tools. The task of protocol keyword inference is a fundamental problem for a wide variety of current and future networking and security services, such as network-based Intrusion Detection and Prevention Systems (IDSes/IPSes), network traffic measurement, network monitoring, and Quality-of-Service (QoS) [2–4]. Take IDSes/IPSes as an example. The protocol parsers of traditional IDSes/IPSes match the packet payload against the protocol signatures represented by a set of regular expressions for traffic management and control. However, this coarse-grained signature checking faces two key challenges: 1). The protocol keywords in the protocol parsers may be incomplete; 2). Traditional protocol parsers lack the abilities of self-learning and continual-learning, which can continually exploit new protocol data from unclassified Internet traffic. Thus, modern IDSes/IPSes need more advanced parsers to implement their functionalities. Besides IDSes/IPSes, protocol traffic analysis is also important for an in-depth understanding of protocol specifications, such as protocol format reverse engineering and protocol state machine inference. The explosive growth of emerging network protocols, such as Peer-to-Peer (P2P) protocols and mobile applications, has raised a number of concerns for security and network management [5]. Thus, to conduct fine-grained network-based protocol specification inference, we first need to classify the protocol traces more accurately.

1.2. Related work and their limitations

The deep understanding of protocol traffic is a core problem regarding network management. Interests in this field have lasted for more than 10 years, and the continuous evolution of protocol traffic analysis justifies this lasting interest. Several elegant approaches

☆ The preliminary version of this paper entitled "Rethinking Robust and Accurate Application Protocol Identification: A Nonparametric Approach" was published in the proceedings of the 23rd IEEE International Conference on Network Protocols (ICNP), San Francisco, CA, USA [1]. This work was supported by the National Natural Science Foundation of China under Grants No. 61402472.

* Corresponding authors at: Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
E-mail address: wangyipeng@iie.ac.cn (Y. Wang).

http://dx.doi.org/10.1016/j.comnet.2017.09.006
1389-1286/

OFFSET	PACKET PAYLOAD							
1–8:	0xc9	0x69	0x7b	0xbc	0x53	0xb1	0x02	0x00
9–16:	0x00	0x0c	0x01	0x00	0x00	0x38	0x32	0xa9

Fig. 1. Example PPLive message for the first 16 B.

have been proposed in the literature, such as Ma's work [6], Discoverer [7], KISS [8], Veritas [9], ProDecoder [10], ProWord [11] and Securitas [12]. The most recent and relevant works are Securitas and ProWord proposed by Yun et al. and Zhang et al, respectively. However, the existing solutions have three major limitations.

First, the completeness of protocol keyword inference for prior literatures is limited by the diversity of protocol behaviors observed in the given trace. For example, considering an SMTP packet "STARTTLS", if this protocol keyword never appears in the given trace for training, oracle for online classification cannot make a right distinction to such network packets. Furthermore, even for some application protocols with available binary executable code, it is still challenging to go through all the protocol states defined in the protocol specifications. Thus, the protocol traces used for offline training dictate the quality of protocol keywords generated by the existing approaches. Secondly, existing work is highly sensitive to statistics. For example, recently *n*-gram approaches have been widely used in the area of networking and network security, and achieved great experimental results in intrusion detection [13] (such as anomaly detection), protocol specification inference [10], and protocol identification [12]. An *n*-gram is considered as a subsequence of *n* elements contained in a given sequence of at least *n* elements. For example, treating each character as an element, the 3-grams generated from message ''RCPT TO" are "RCP", "CPT", "PT_", "T_T", and "_TO". Note that even for a modest value *n* (i.e., 3), the number of *n*-gram elements is tremendous, where the state space of 3-grams involves 16 million (256^3) unique elements. Thus, in practice, the *n*-gram approaches usually select a subset of *n*-grams with high probability of appearance in the protocol trace, and directly use the selected subset as the building blocks for deep packet analysis. However, this naive solution has serious disadvantages. *n*-grams, that are statistically insignificant in the given trace but belong to true protocol keywords, are ignored in the analysis. In addition, substring-based methods, such as ProWord [11], are also sensitive to statistics. ProWord counts the occurrence probability of a byte or subsequence to identify field boundaries in protocol packets. Consider the example PPLive packet in Fig. 1, where Offset 5 (i.e., byte "0x53") is a command field of PPLive, denoting a request announcement from an active peer. In reality, bytes "0x52", "0x53", "0x54", "0x55", and "0x56" at offset 5 are all protocol commands for PPLive. We notice that the occurrences of byte "0x55" at offset 5 are statistically insignificant in the trace of PPLive, Thus, network packets with such a command are ignored in the previous state-of-the-art solutions. In summary, statistics dependency methods decrease the performance on recall and lead to inaccurate results for network-based protocol identification. Thirdly, the computational efficiency of the prior literatures, such as running speed and memory requirement, represents one of their limitations. For example, we notice that Securitas [12] needs 6–7 ms to process a network packet on a cluster machine running at 2.13 GHz. The main reason is that the feature extraction process of Securitas for online classification depends on a Markov chain Monte Carlo sampling scheme, which needs more computational cycles to recover the correlations among the *n*-grams. Thus, we find prior literatures impose more restrictions to generate a protocol keyword. To sum up, in this paper we aim to address the aforementioned limitations of the existing approaches, and provide more accurate results for network-based application protocol identification.

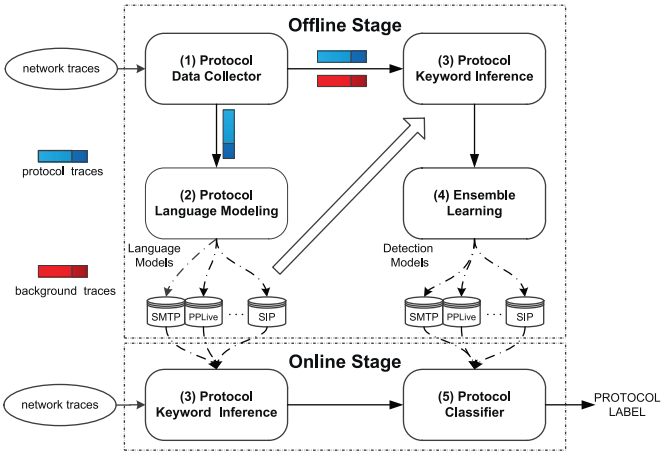

Fig. 2. Architecture of ProHacker.

1.3. Proposed approach

In this paper, we propose ProHacker, a nonparametric approach that automatically infers robust and accurate protocol keywords from the packet payload of network traces and effectively identifies the protocol trace from mixed Internet traffic. ProHacker regards application protocols as languages for application softwares, in the sense that they both use well-defined formats for data exchange between hosts, and to reach agreement, the technical specifications are agreed on by all the parties involved. Our approach is based on the key insight that the *n*-grams of protocol traces, just like words of natural languages, have highly predictable statistical nature that can be effectively captured by statistical language models and be leveraged for robust and accurate application protocol identification.

The key novelty of ProHacker lies in its solution of the zero probability problem using a smoothing technique for *n*-grams that were not observed in the selected subset for protocol keyword inference. Furthermore, ProHacker addresses the aforementioned three limitations of prior work. In practice, ProHacker is based on a comprehensive statistical analysis of packet payload, and thus it requires no knowledge of protocol specifications as a prior. Fig. 2 shows the architecture of ProHacker. ProHacker has five major modules, (1) Protocol Data Collector, (2) Protocol Language Modeling, (3) Protocol Keyword Inference, (4) Ensemble Learning and (5) Protocol Classifier, for Offline Stage and Online Stage. We next give a brief description of the two stages.

1.3.1. Offline stage

This stage aims to build two kinds of models for each target protocol – language models and detection models. First of all, (1) Protocol Data Collector filters mixed Internet traffic in order to label packet traces of different application protocols. We give a concrete description to our strategy of this module in Section 5.1. Next, using the collected protocol traces, (2) Protocol Language Modeling builds a language model for the bytes sequences generated by an application protocol based on a hierarchical Pitman–Yor process [14,15]. This module enables ProHacker to derive approximate estimates for *n*-grams that were not observed in the selected subset for protocol keyword inference.

Secondly, using the protocol language models, (3) Protocol Keyword Inference extracts the robust and accurate protocol keyword associated with each packet as its features. Next, (4) Ensemble Learning uses a small amount of labeled network packets and a large amount of unlabeled network packets to train a protocol classifier for each target protocol. Using the corresponding proto-