Computer Communications 84 (2016) 40–51

Contents lists available at ScienceDirect

Computer Communications

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

Towards automatic protocol field inference

Ignacio Bermudez[a,*], Alok Tongaonkar[a], Marios Iliofotou[b], Marco Mellia[c], Maurizio M. Munafò[c]

[a] Symantec Corporation, 350 Ellis St., Mountain View, CA 94086, USA
[b] Caspida, 2100 Geng Road#100, Palo Alto, CA 94303, USA
[c] Politecnico di Torino, Corso Duca degli Abbruzzi 24, Turin 10129, Italy

ARTICLE INFO

Article history:
Received 21 September 2015
Revised 20 February 2016
Accepted 23 February 2016
Available online 3 March 2016

Keywords:
Protocol reverse-engineering
ICS/SCADA Networks
Payload based anomaly detection

ABSTRACT

Security tools have evolved dramatically in the recent years to combat the increasingly complex nature of attacks. However, these tools need to be configured by experts that understand network protocols thoroughly to be effective. In this paper, we present a system called *FieldHunter*, which automatically extracts fields and infers their types. This information is invaluable for security experts to keep pace with the increasing rate of development of new network applications and their underlying protocols. FieldHunter relies on collecting application messages from multiple sessions. Then, it performs field extraction and inference of their types by taking into consideration statistical correlations between different messages or other associations with meta-data such as message length, client or server IP addresses. We evaluated FieldHunter on real network traffic collected in ISP networks from three different continents. FieldHunter was able to extract security relevant fields and infer their types for well documented network protocols (such as DNS and MSNP) as well as protocols for which the specifications are not publicly available (such as SopCast). Further, we developed a payload-based anomaly detection system for industrial control systems using FieldHunter. The proposed system is able to identify industrial devices behaving oddly, without any previous knowledge of the protocols being used.

1. Introduction

In recent years attacks against networks have become more complicated. To defend against these complex attacks, network security systems have also evolved to use more sophisticated mechanisms. For instance, firewalls have moved from using simple packet-filtering rules to using application level rules that need deeper understanding of the protocols being used by network applications. Similarly, intrusion detection systems are increasingly using vulnerability based signatures [1] that contain information specific to network protocols. Access control mechanisms are also evolving from IP address based policies to fine-grained policies which use protocol objects such as users and message types.

It is clear that configuring all of the above applications requires a deeper understanding of the network protocols, which is done through reading protocol specifications. However, comprehending protocol specifications is a very tedious task. Moreover, many of the proprietary protocols specifications are not publicly available. The traditional approach of manual reverse engineering a protocol cannot cope with the rate at which new benign or malicious applications are made available and brought into workplace. As a result, security administrators have to configure security applications with very limited visibility into the network protocol space; thus adversely affecting the efficacy of these tools in securing the network.

The above technology challenge has led to a growing interest in the research community in the development of techniques for automating the reverse-engineering process for extracting protocol specifications, which consists of inferring message formats and underlying protocol state machines. The state-of-the-art techniques can be classified in two categories: reverse-engineering through binary code analysis [2–6] and from network traffic [7–13]. In this work, we present an automatic reverse-engineering system of the second category, i.e. it infers protocol specifications from just network traffic data. Reverse-engineering using network traffic has an advantage over techniques using binary analysis, because application binaries are not always available to the security operators.

Our approach to the problem of protocol reverse engineering aims to extract field boundaries and field protocol types from network traces that belong to the protocol. As compared to previous

* Corresponding author. Tel.: +1 4084665710.
E-mail addresses: ignacio_bermudezcorr@symantec.com (I. Bermudez), alok_tongaonkar@symantec.com (A. Tongaonkar), marios@caspida.com (M. Iliofotou), marco.mellia@polito.it (M. Mellia), maurizio.munafo@polito.it (M.M. Munafò).

http://dx.doi.org/10.1016/j.comcom.2016.02.015
0140-3664/© 2016 Elsevier B.V. All rights reserved.

works in this area, we are able to extract richer protocol information in terms of (i) extracting diverse field types, and (ii) handling binary and textual protocols in an uniform framework. We study well known protocols and identify a set of field types that can be used in a multitude of security applications. We focus on identifying: (i) Message Type (MSG-Type), such as flags in DNS protocol or GET/POST keywords in HTTP, (ii) Message Length (MSG-Len), usually found in TCP protocols to delimit application messages in a stream, (iii) Host Identifier (Host-ID) such as Client ID and Server ID, (iv) Session Identifier (Session-ID) such as cookies, (v) Transaction Identifier (Trans-ID) such as sequence/acknowledgment numbers, and (vi) Accumulators such as generic counters and timestamps. We note that a protocol may not have all the above types of fields.

We built a system called FieldHunter, that uses a two step methodology: (i) Field extraction: here we extract fields from the protocol messages. (ii) Field type inference: here we infer the type of the fields extracted in the previous step. The key contribution of our work is the development of various heuristics based on observed statistical properties for inferring the different field types. In our evaluation, we used real network traces from three different Internet Service Providers (ISPs) to validate the ability to extract various field types from well known protocols such as Real Time Protocol (RTP), as well as protocols without any publicly available specification such as SopCast's protocol.

Next, to illustrate the use of FieldHunter in building end-to-end security applications, we developed a payload-based anomaly intrusion detection system for industrial control systems. Industrial Control Systems (ICS) encompass several types of control systems used in industrial production, including Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other smaller control system configurations such as Programmable Logic Controllers (PLC). Due to their use in the industrial sectors and critical infrastructures, they are a prime target for attackers and the cost of successful attacks is tremendous to the victims. Typically, the attackers targeting ICS are sophisticated, and have a lot of resources; sometimes even being state sponsored. They are able to avoid detection by traditional defense mechanisms [14]. To make matters worse, there has been a trend of increasing number of attacks on critical infrastructure as evidenced by the rapid increase in the number of reported attacks on ICS from 91k in 2012 to over 675k in 2014 [15]. To the best of our knowledge, our system is the first payload-based anomaly detection system that handles legacy proprietary protocols commonly used in ICS networks.

The rest of the paper is organized as follows. Section 2 defines the terminology used throughout the paper, Section 3 provides details about the core algorithms used by FieldHunter. Performance evaluation and parameter tuning are presented in Section 4. We describe the anomaly detection system for ICS in Section 5. We discuss about assumptions and limitations in Section 6, related works in Section 7 and finally conclude this work in Section 8.

2. Terminology

Fig. 1 shows a pictorial representation of the terminology used throughout this work. Our system uses as input a set of *conversations* [1] of a particular application. We refer to such a set as *collection*. Conversations consist of exchanged *messages* between two hosts. Messages from client to server are denoted as C2S (dark-colored) and from server to client as S2C (light-colored). We consider the initiator of the conversation as the client, the other end as the server, and identify hosts by their IP address. Messages consist of different pieces of information enclosed in *fields*. As we show in Fig. 1, conversations evolve horizontally over time (t) and messages can be compared vertically across multiple conversations.

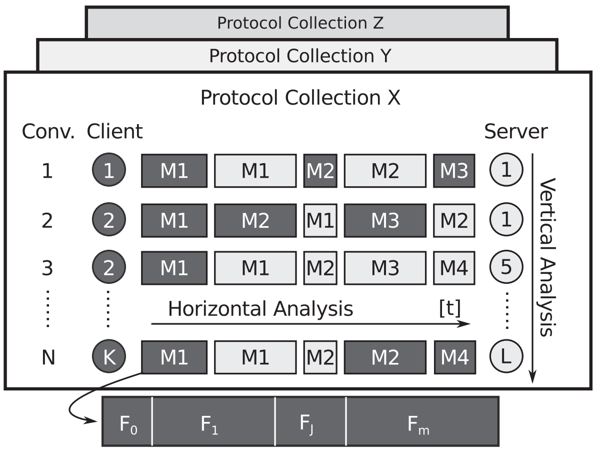

Fig. 1. Terminology diagram.

To enable the analysis of a collection, the messages in the conversations can be grouped together in the following ways: (i) Grouping messages based on their position in conversations, e.g., all third messages in C2S direction. (ii) Grouping together all the messages of a conversation. This essentially captures session-like information. (iii) Grouping together messages by direction, e.g., all C2S messages. We note that (i) and (ii) are very similar to vertical and horizontal sub-collections as defined by Kreibich et al. [16]. Message grouping is instrumental for FieldHunter to find patterns in the collections. If these groups do not contain enough message diversity, FieldHunter cannot unveil the field types it is designed for.

It is worth mentioning that the formation of protocol collections used by FieldHunter is beyond the scope of this work. However, we suggest two alternatives for the same. One way is to use a test-bed in which the application is executed while the traffic exchanged is being captured. Alternatively, the collection can be extracted from passive observation of actual traffic by the means of network classifiers, i.e., by filtering all conversations involving a well-known port (see Section 5), or by relying on a behavioral traffic classifier classifier [17].

Application conversations are transported by TCP/UDP segments and are extracted by FieldHunter using the following methodology: (i) for messages transported over UDP it is assumed that each segment contains one application message, and (ii) for TCP it is assumed that TCP PUSH flags delimits the beginning of a new application message from the end of another one. An accurate message extraction can be done once the MSG-Len field has been identified by FieldHunter.

We make a distinction between textual and binary protocols as follows: Textual protocols use human readable words and symbols to structure data, and they look more like a text document. Exponents of textual protocols are HTTP and SMTP. On the other hand, binary protocols encode data on bits rather than symbols and the way that data is structured is quite rigid. Examples of binary protocols are DNS and DNP.

3. Design

In this section, we describe the system design and discuss the two components of FieldHunter(i) Field Extractor (ii) Field Type Inference Engine. These components are run in sequence to obtain

[1] A conversation is formed of the two flows in opposite directions, where a flow is defined by the 5-tuple (Layer-4 Protocol, Source IP, Source Port, Destination IP, Destination IP).