Available online at www.sciencedirect.com
ScienceDirect

ELSEVIER

December 2013, 20(Suppl. 2): 75–79
www.sciencedirect.com/science/journal/10058885

The Journal of China Universities of Posts and Telecommunications

http://jcupt.xsw.bupt.cn

Protocol reverse engineering through dynamic and static binary analysis

WANG Ying[1] (✉), GU Li-ze[1], LI Zhong-xian[2], YANG Yi-xian[1]

1. Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. National Cybernet Security Ltd, Beijing 100088, China

Abstract

This paper presents a new method for protocol reverse engineering, which combines both the dynamic and static binary analysis. Our work not only does precise positioning on the field and its length, but also gives the field attributes accurately. According to different instructions and the current program structure, we can infer the message format validly. To prove the method is sound and effective, we build a prototype tool – NetProtocolFinder, and select some documented protocol and undocumented protocol messages as the test instances respectively. Results of our experiments show that the tool can not only extract the message format from protocols effectively, but also speculate the state machine model through relevant field attributes conveniently.

Keywords automatic protocol reverse engineering, dynamic binary analysis, static binary analysis

1 Introduction

Protocol reverse engineering is the technology of extracting the specific documented or undocumented protocols. Having a detailed knowledge of the special protocol may do great help for many security applications such as fuzzing test [1], deep packet inspection [2], network-based intrusion detection systems [3] and botnet counter strategy [4].

Due to the importance, protocol reverse engineering projects have been researched for many years. Those past works relied on manual techniques, which usually were complex and time-consuming. To change this situation, a large number of works have been introduced to solve this problem. To address this problem, automatic protocol reverse engineering techniques [5–6] have been proposed to automate the process of reverse engineering network protocols.

The protocol informatics (PI) project [7] is the representative of the sequence alignment method. Another method is using data mining theories [8–9]. The method of dynamic binary analysis in Refs. [10–11] is also introduced to solve this problem. Additionally, another method named context-aware execution monitor method has been conceived for this task.

The work of Ref. [12] was the first one led into the dynamic binary analysis method, but their work can only identify a few field attributes. In Ref. [13], Wondracek et al. improve the dynamic binary analysis method through proposing the combination of the dynamic binary analysis and the data mining method. The main contribution of Ref. [14] was to extract the encrypted message, but this method does not apply in some cases. Based on the previous works, Ref. [15] was proposed to infer the message format exactly. They introduced the identifying method of the send message. Besides this, they improved the method of dealing with the encrypted message.

However, it is mostly a time-consuming and tedious process to derive protocol specifications. For the undocumented protocol, the specification even has to be reverse engineered manually. Traditionally, this task is a complex work for many reasons: a single protocol message usually contains a large number of fields; an individual field may not be static and may have varying size; some fields may depend on each other very closely and so on.

Received date: 02-11-2013
Corresponding author: WANG Ying, E-mail: wangying_bupt@126.com
DOI: 10.1016/S1005-8885(13)60217-4

In this paper, a tool named NetProtocolFinder is specifically designed to verify our method. We present a new approach based on dynamic binary analysis and static binary analysis in our system. We take some documented and undocumented protocols for our test instances, respectively. The result shows that our method can extract field attributes effectively and the state machine model can be obtained conveniently.

This paper is organized as follows. Sect. 2 describes our improved method of dynamic and static binary analysis. Sect. 3 gives the testing results of our tool. Sect. 4 is about the limitations of our work. Sect. 5 concludes the paper.

2 Approaches and system design

2.1 Approaches

We find that the combination of dynamic binary analysis and static binary analysis can identify the field attributes more effectively than using only the dynamic binary analysis method. Thus, before discussing our methods, some algorithms of static binary analysis are briefly explained as follows:

Loop-detection algorithm: the loop is one of the based structures of a program. A lot of research has been done on this subject of loop detection. The major two algorithm of this detection are interval finding algorithm and identifying loops using DJ graphs [16]. In our system, we take the second one for further accurately.

Switch-detection algorithm: the switch is another based structure of a program. As we all know, this one cannot be easily identified using some existed algorithms. Therefore, we introduce a new rule made of some specific instructions. If there are some instructions belonging to this rule, we can confirm that the current structure is a switch. This rule relies on the current instructions. First, we filter the operator and its first operand of the instruction. If the operator is decrease (DEC) , substract (SUB), compare（CMP）, TEST and the first operand is related with the taint memory address, we can take a further step to judge the current structure. Then we find the conditional jump (JMP) in the next instruction. If the condition is met, we called it ‘match’ once. When we can take this procedure of finding ‘match’ for at least twice, the current structure is determined as a switch. Fig. 1 illustrates the core algorithm of our approach.

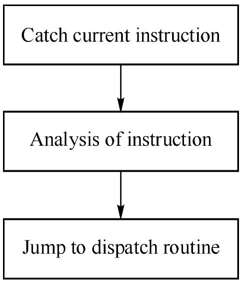

Fig. 1 Core algorithm

From the Fig. 1 we can find the core algorithm is comprised of three parts. The first is the procedure of getting current instruction. We can get it from the target process using our disassembly engine. The second one is acquiring current instruction semantic by its opcode. Then the core can enter into different dispatch routines according to different opcode. As described above, the third part is composed of many dispatch routines. Unfortunately not every instruction has the related dispatch routine. Some of them will enter the default routine, because we do not care the opcode. We can assume that $T=Ls+o$ under one of the two encodings, where T and L are the length of the target field and the value of the length field respectively, and s is scale, o is offset. s and o are two integer constants [13]. When merging two length fields a and b, we can compute Eq. (1):

$$\left.\begin{aligned} s &= \frac{T_a - T_b}{L_a - L_b} \\ o &= T_a - T_a s = T_b - T_b s \end{aligned}\right\} \qquad (1)$$

We skip the unconcerned instructions. Then we can start to infer the field attributes according to the arguments of the current instruction. We introduce the method of inferring the dynamic field attribute at first. When an instruction contains the operator add （ADD), we should focus on its two arguments. If one argument is the taint memory address and the other argument is the content of another taint memory address, we can make a decision that this field is a dynamic field. However, this dynamic field is a pointer field or a length field, which is determined by the taint memory address offset. Generally, if the taint memory address is the beginning address, we can extract this field as a pointer field. Otherwise, it is a length field.

One mechanism of getting the static field attributes is detecting the corresponding instructions. For example, once we find that the current instruction contain the operator compare (CMP) and the specific arguments, this means the field attributes may be one of three types. If it is a code field, the instruction should belong to a switch structure.