Digital Investigation 16 (2016) 19–28

Contents lists available at ScienceDirect

Digital Investigation

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

Determining image base of firmware for ARM devices by matching literal pools

Ruijin Zhu [a], Yu-an Tan [a], Quanxin Zhang [b], Yuanzhang Li [a], Jun Zheng [a, b, *]

[a] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China
[b] Research Center of Massive Language Information Processing and Cloud Computing Application, Beijing, 100081, China

ARTICLE INFO

Article history:
Received 1 August 2015
Received in revised form 19 January 2016
Accepted 21 January 2016
Available online 10 February 2016

Keywords:
Image base
Literal pool
Reverse engineering
Firmware
ARM

ABSTRACT

In the field of reverse engineering, the correct image base of firmware has very important significance for the reverse engineers to understand the firmware by building accurate cross references. Furthermore, patching firmware needs to insert some instructions that references absolute addresses depending on the correct image base. However, for a large number of embedded system firmwares, the format is nonstandard and the image base is unknown. In this paper, we present a two-step method to determine the image base of firmwares for ARM-based devices. First, based on the storage characteristic of string in the firmware files and the encoding feature of literal pools that contain string addresses, we propose an algorithm called FIND-LP to recognize all possible literal pools in firmware. Second, we propose an algorithm called Determining image Base by Matching Literal Pools (DBMLP) to determine the image base. DBMLP can obtain the relationship between absolute addresses of strings and their corresponding offsets in a firmware file, thereby a candidate list for image base value is obtained. If the number of matched literal pools corresponding to a certain candidate image base is far greater than the others, this candidate is considered as the correct image base of the firmware. The experimental result indicates that the proposed method can effectively determine image base for a lot of firmwares that use the literal pools to store the string addresses.

Introduction

Embedded devices have become the usual presence in our life, such as cell phones, digital cameras, printers, smart watches and so on. All these devices run special software, often called firmware, which is usually distributed by vendors as firmware updates or firmware images (Costin et al., 2014). Firmware is the soul of embedded devices, because some embedded devices have no other software besides firmware, and the firmware also determines the function and performance of the device.

In the field of digital forensics, we need to reverse analysis firmwares of the embedded devices in some scenarios. For example, (1) By reverse engineering the firmware, back door has been discovered in some devices, such as D-Link (Heffner, 2013a) and Schneider Electric Quantum Ethernet Module (Santamarta, 2011). (2) If data stored on the devices is encrypted, reverse engineering can be applied to obtain the encryption algorithm and even the encryption key which can recover the clear data (Zhang et al., 2015). (3) By reverse engineering the firmwares released by the competing companies, we can determine whether they plagiarize our company's algorithms or infringe our patents and so on. Hence, reverse engineering is an important technology in the digital forensics.

* Corresponding author. School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China.
E-mail addresses: ruijinzhu@gmail.com (R. Zhu), tan2008@bit.edu.cn (Y.-a. Tan), zhangqx@bit.edu.cn (Q. Zhang), popular@bit.edu.cn (Y. Li), zhengjun_bit@163.com (J. Zheng).

http://dx.doi.org/10.1016/j.diin.2016.01.002
1742-2876/

Reverse engineering takes a software system as input and uses some technology (such as disassembling and system analysis) to deduce the software source code, design principles, application structures, algorithms, operation processing and related documentation. Reverse engineering not only can avoid duplicating efforts and improve the efficiency and quality of software, but also can translate legacy system into evolution system to efficiently reuse them. Some tools such as Binwalk (Heffner, 2013b), avatar (Zaddach et al., 2014), FRAK (Cui et al., 2013) and BAT (Hemel and Coughlan, 2009) have been designed to modularize the firmware unpacking, modification and repacking processes. They are particularly useful in reverse engineering of firmware. By utilizing reverse engineering, some previously unknown vulnerabilities or security weaknesses have been discovered in banking application (Yoo et al., 2015) and some firmwares of devices, such as SSD (Zhang et al., 2015), printer (Cui et al., 2013) and satellite phone (Driessen et al., 2012).

In reverse engineering area, when disassembling executable file, disassembler needs to know processor type of its runtime environment and image base[1] of executable file (Basnight et al., 2013a). For a given embedded system firmware, we can easily get the processor type[2] but cannot get the image base of firmware. Correctly setting the image base in disassembler during the initial import enhances the analysis of the firmware. More specifically, setting the correct image base ensures that subsequent cross references are accurate where the cross references use absolute addresses rather than offsets in the firmware (Schuett et al., 2014). Cross references include code cross-references and data cross-references. When lack of these cross references information, we are difficult to navigate efficiently in disassembly listing. Facing the obscure disassembly code, people often lost their direction when they look for parts code that they are most interested in. On the other hand, knowledge of the correct image base is critical in understanding the firmware as a whole. Working with an incorrect image base may lead to inaccurate interpretations of segments referenced by absolute addresses (Basnight et al., 2013b).

Related work

As reverse engineering of firmware develops, people have put a great deal of effort into determining the image base of firmware techniques. Skochinsky (2010) proposed a general principle for determining the image base of file with unknown format. They suggested some kinds of hints, such as self-relocating code, initialization code, etc., can be used, but it is not an automatic method and heavily relies on the engineers' experience. Basnight et al. (2013b) presented an overview of the reverse engineering process and proposed a method which can analyze to learn the image base by absolute addresses in the instructions. Heffner (2011) presented a method to infer image base from decompress code in firmware. Utilizing zeroing loops code in BSS section, Heffner (2015) also inferred the image base of firmware file. Peck et al. (2009) scanned the firmware image to look for zlib compressed section in which they found some symbol names. When looking through the firmware, a very regular ten-byte pattern was found in some offset. They speculated these ten-bytes are addresses of symbol names, thereby inferred image base.

Santamarta (2011) mentioned that it needs some tricks to determine image base of firmware, and introduced two methods. The first method is using the “li instructions trick” for MIPS firmware. This method consists of searching the li instructions that load an absolute address into a register. The trick presumes that a significant number of absolute addresses refer to locations in the firmware itself, and therefore have the same base address. Candidate image base are then tested by rebasing the firmware and determining if the absolute addresses correctly align with target data such as functions or strings. Then we can find the image base with trial and error. The second method is to use absolute addresses in jump table to determine image base. The jump table is comprised of absolute addresses of cases, and then the distances between the cases are calculated. If a certain distance is different from others, the corresponding relation between the absolute address of case and offset can be obtained, by which the base address can be determined.

All the above methods require intuition and experience of reverse engineer; in other words, the success and effectiveness often rely on the human factor. And none of them focused on automatic methods that can determine the image base of unknown format files.

Contributions

Binary files with unknown base in reverse engineering mostly come from embedded firmware of which about 63% are based on ARM devices (Costin et al., 2014). Hence, we focus on the image base of firmware on ARM-based devices and propose a new method to automatically determine the image base in this paper. The main contributions of our work are summarized as follows.

(1) Based on the characteristics of literal pool, we propose an algorithm called FIND-LP to recognize all possible literal pools that contain string addresses in firmware. Besides, we can get the string information including string lengths and offsets in the firmware.

(2) We propose an algorithm called Determining image Base by Matching Literal Pools (DBMLP) to determine image base of firmware for the first time. The algorithm utilizes the literal pools recognized by FIND-LP and the string information to calculate some appropriate memory locations which are candidate image base. If

[1] The image base is the base address of the executable file loaded into the memory.

[2] There are usually several ways to get processor type, such as consulting the product manual (Basnight et al., 2013a), physical examination of the device (Basnight et al., 2013b), disassembling the firmware and guessing the processor type (Basnight et al., 2013a).