Tools and Strategies for Reverse Engineering the Format of Statistical Data Files

Matt Shotwell

Vanderbilt University

1 Introduction

Reverse engineering is much like forensics. At some point in the past, the process was conceived and implemented. In the present, the input and output are observable, but the process itself is a *black box*, that is, the details of its design and implementation have been lost, or concealed. Forensics, and reverse engineering attempt to recover the details.

The risks of a reverse engineering venture may be considerable. Indeed, the principal risk is failure to recover the details of a process under study. This is compounded by the consequences that may arise regardless of success or failure. For instance, merely the attempt to reverse engineer certain computer software may be in violation of the associated end-user license agreement. In an extreme example, an attempt to "crack" encrypted communications may be illegal.

2 Prerequisites

This discussion assumes familiarity with some modern computer concepts, such as file input and output, and how computers store data in bits and bytes. References are given throughout for further reading on key concepts.

Statistical data files are regular computer files that contain structured data, such as a table of records and fields. A data file may also contain metadata, such as a record count or field labels. A file *format* is a specification that determines how structured data and metadata are organized into a computer file. Logically, a computer file is simply a sequence of eight bit binary values(bytes). Hence, a file format describes how structured data (*e.g.* integers, text) are (1) represented and (2) serialized in a sequence of bytes.

Formatted data may be human readable, that is, consisting of bytes that are interpreted as character strings (*e.g.*, comma separated values). Encoded data that are not human readable are generically said to have 'binary' formatting (*e.g.,* XBase and dBase formats, **?**).

3 Basics

4 The Knockout Strategy

5 Deducing Field Width Using Endianness

When a binary field encodes a multi-byte quantity, it may not be clear how many bytes contribute to the value. For instance, suppose that a multi-byte, unsigned integer field is suspected to be four bytes in length, but is only observed for values less than or equal to $2^{16} - 1$. In this case, it is possible that the field is only two bytes in length, and the remaining bytes constitute a separate two-byte field.

If the suspect field is subsequently observed in the opposite endianness, the field length becomes clear. To illustrate, consider the four bytes (in hexadecimal representation) `01 00 00 00`. Then suppose we observe the same field in opposite endianness. There are several possibilities: 1) `01 00 00 00` - the field is single byte, `00 01 00 00` - the field is two-byte, `00 00 01 00` - the field is three-byte, and `00 00 00 01` the field is four-byte.

If there is concern that bytes at a particular offset may form incomplete parts of adjacent fields, then this test for field withd may be misleading. If the byte values observed in opposite endianness cannot be obtained by reordering the original byte values, then these bytes must span two adjacent fields.

6 Legality

Some material from: https://www.eff.org/issues/coders/reverse-engineering-faq

Among intellectual property laws, including Copyright and fair use (17 U.S.C. 107), DMCA (17 U.S.C. section 1201), trade secrets (Uniform trade secrets act with amendments 1985), Contract law (EULA, TOS, TOU, NDA, developer or API agreement), Electronics Communications Privacy Act (18 U.S.C. 2510 et.seq.), trade secret law is most directly impacts reverse engineering of statistical data files (since copyright (of code), cracking, are not involved, but be careful about TOS and EULAs). However, the UTSA makes explicit that reverse engineering is a *proper* means.