Converting String, Hexadecimal and Fix Numbers from Ruby Perspective

By | Mohd Hafiz Mat Tabrani

Introduction

Software development for security domains has always involved converting from and to hexadecimal and binary formats especially for Malware Analysts. For those new to certain languages, a high learning curve is involved here and this inevitably translates to high development cost.

There are many tools out there that can perform the conversion from one format to another. While some of them they are easy to use, but for those Malware Analysts who needs to process tons of binaries using their self-developed scripts that tailor specifically to their needs, would find it difficult, if not impossible, to integrate those tools in their scripts.

This article concentrates on the use of the ruby language since it is among the favourite language to the new comers to help them shorten the learning curve. Even though the fundamental knowledge how data store in memory will help others to strengthen their understanding of data representation throughout memory.

To help us understand this, we will use the data below: str = "ABC D"

Please note that there is a new line character (\n) between C and D. Table 1.0 contains the same variable presented in four different formats.

Numbering Format	Data				
String(Binary)	A	B	C	\n	D
Hex	41	42	43	0A	44
Fixnum(Decimal)	65	66	67	10	68
Binary	01000001	01000010	01000011	00001010	01000100

Table 1 *Data Presentation Comparison*

When store 'A' character into a variable, it needs to be placed in memory. Since RAM can only store 1 and 0, the 'A' character needs to be converted to binary format. Base on ASCII table (http://www.asciitable.com/) it is agreed that the 'A' character should have 01000001 which is equivalent to 65 in decimal.

From the ASCII table, a new line character (\n) will be stored as 00001010 in RAM which is equal to 10(decimal).

Now, we are ready for the next phase which is to convert the data into the ruby language. In this example, we will show you how to convert data from and to Hex-BinaryString-Binary where these are commonly used in software development for the information security industry.

Converting Hex to BinaryString

First we will look at how to convert hex to binary.

sHex = "4142430A44"
puts [sHex].pack('H*') ==> "ABC\nD"

pack() is a method for array object. Originally sHex is a string, so we need to put it in the block to convert it to array.

Pack method will produce a BinaryString. The 'H*' directive will tell ruby that the array element is a Hex string.

There are many directives available (http://ruby-doc.org/core/classes/Array.html#M002222).

Converting BinaryString to Hex

For converting BinaryString to hex, we should use unpack with H* as the format parameter.

str = "ABC\nD"
str.unpack('H*') ==> ["4142430A44"]

unpack() will return array which contains a string of the hex format in its first element. To get the string of hex you can try:
str.unpack('H*')[0] ==> "4142430A44"

Converting Binary String to Binary

Binary is data stored using either 1 or 0. This format is used by computers to store data in memory chips or hard disks. To unpack(), one can also be used to present data in binary. Use B* as the format parameter as below:
str.unpack('B*')[0] => "01000001010000100100001100001010001001000100"

The result is quite long. To understand it, split the string so that each group has eight numbers. This is because each character consumes eight bit in memory.

01000001	01000010	01000011	00001010	01000100
A	B	C	\n	D

Table 2 *Data Presentation Of Binary and Ascii*

Converting Hex to Binary

"41".class is a String. This means our memory will store "00110100" (decimal =52, hex = 34) and "00110001"(decimal = 49, hex = 31).

"41".hex.class is a FixNum. "41".hex will tell ruby to read those string as hex, as a result stores "01000001" (decimal = 65, hex = 41) in memory. The two examples will definitely be interpreted differently by a CPU.

To display the same value in binary we can use the to_s(2) method from the Fixnum class.

"41".hex.to_s(2) ==> "01000001"

The result is a string which contains a binary representative of 0x41.

Value 2 for the parameter means to display the value in base 2. Sending 16 as base will output the same result "41", as hexadecimal is base 16. You can try to pass any integer between 2 and 36 and study the output for further exercise.

Converting Binary to BinaryString

Pack('B*') method from Array class will process the first element of the array and present it in BinaryString.

["0100000101000010010000110000101001000100"].pack('B*') ==> "ABC\nD"

Conclusion

Always remember, that machines do store information in streams of 0 and 1. Since human have limitations in memorizing long numbers, hex representation is used which can still represent the same value.

Different from both above, string is a stream of character human use for storing information. ASCII table is used to convert information stored in a computer to a format that humans can understand.

I hope this article will help programmers realise that ruby provides very powerful built-in features related to binary processing. Do learn and understand them, and use it to develop more robust and in the same time sensitive tools. ■

References

1. *http://ruby-doc.org/core/classes/Fixnum.html#M001050*
2. *http://ruby-doc.org/core/classes/Array.html#M002222*
3. *http://ruby-doc.org/core-1.8.7/classes/String.html#M000689*
4. *http://en.wikipedia.org/wiki/ASCII*