Writeup by Razvi
RazviOverflow Twitter Github

Integer Operations

February the 20th 2020
Last edited on February 23, 2020

Contents

1. Introduction

Disclaimer: *I assume you solved or at least tried to solve the binary before reading this document. Its main purpose is purely educational and it is intended to help you learn and further your knowledge.*

In this first chapter we will see what this binary is all about and introduce the main concepts needed in order to solve it.

1.1 Challenge description

Title: Integer Operations

Description: You are back to school learning about addition, subtraction and equations with integers. You'll have to fully understand how they behave in order to pass the exam.

The name of the binary/challenge as well as its description are a good hint and starting point. *Integer operations* and *how integers behave* are clear references to the well-known vulnerability/software bug ***Integer Overflow***.

1.2 Integer Overflow

There is a lot of info about integer overflows/underflows, why the occur in memory, how to exploit them, how to avoid them, etc... But I consider the following references as the main ones to start understanding it:

1. Basic Integer Overflow by Blexim. Published in Phrack#60 [1].
2. Understanding Integer Overflows in C/C++ by Dietz, Li, Regehr and Adve [2].
3. Integer Overflow overview autogenerated by Science Direct [3].
4. Integer Overflow entry on Wikiedia [4]. Even though Wikipedia has bad reputation, the entry about Integer Overflows is pretty good and very useful to read discover other references and resources (bottom of the page).

Broadly speaking, integer overflows/underflows occur when a given number is stored in a certain data type that it does not fit into. That is, a number bigger than the maximum integer (`int`) value or a number lesser than the minimum integer value pretended to be assigned to a `int` variable. They can also appear when signed data types are treated as unsigned or viceversa.

Integer overflows are a type of numerical software errors/bugs. "*These errors include overflows, underflows, lossy truncations (e.g., a cast of an int to a short in C++ that results in the value being changed), and illegal uses of operations such as shifts (e.g., shifting a value in C by at least as many positions as its bitwidth).*" [2].

Back in 2002, Blexim categorized these numerical errors into: 1) Integer overflows and 2) Signedness bugs [1]. According to him/her, "*an integer overflow is the result of attempting to store a value in a variable which is too small to hold it*". Two typew of integer overflows were described:

Table 1.1: CWE Weaknesses involving integer and numerical errors. (Source: CWE CATEGORY: INT)

Identifier	**Title**
CWE-190	Integer Overflow or Wraparound
CWE-191	Integer Underflow or Wraparound
CWE-192	Integer Coercion Error
CWE-194	Unexpected Sign Extension
CWE-195	Signed to Unsigned Conversion Error
CWE-197	Numeric Truncation Error
CWE-680	Integer Overflow to Buffer Overflow

- Widthness overflows. They occur when a variable of a given size (width) like `int` which is typically 32 bits long, is assigned to a smaller variable like `short`, which is typically 16 bits long. Blexim also talks about *truncation* and *integer promotion/demotion* that are quite important concepts.
- Arithmetic overflows. They occur when the result of an arithmetic operation is bigger than the data type of the variable it will be stored into.

On the other hand, signedness bugs "*Signedness bugs occur when an unsigned variable is interpreted as signed, or when a signed variable is interpreted as unsigned. This type of behaviour can happen because internally to the computer, there is no distinction between the way signed and unsigned variables are stored.*"

As it was aforementioned, there is plenty of literature about integer overflows. It is a well known vulnerability/bug and, as such, there are plenty of public recommendations to deal with it. The Common Weakness Enumeration[1] (CWE) defines various weaknesses that involve integer overflows or other types of numerical errors. Table 1.1 shows some of these weaknesses (there are many more in the CWE's INT category[2]). Moreover, the weakness Integer Overflow or Wraparound (CWE-190) is placed 8th in the 2019 CWE Top 25 Most Dangerous Software Errors [5] ranking.

[1] `https://cwe.mitre.org/index.html`
[2] `https://cwe.mitre.org/data/definitions/1158.html`

2. Solving the challenge

In order to solve the challenge you will only need:

- Understanding how integer overflows and truncation work in memory (at bit level).
- A calculator with bit-view mode, and preferably bit-toggling capabilities, in order to help you visualize what is actually happening. Windows10's default calculator in Programmer is awesome. As a Linux alternative you can use KCalc[1].

Figure 2.1: Binary's introduction.

Figure 2.2: Formulation of the first level.

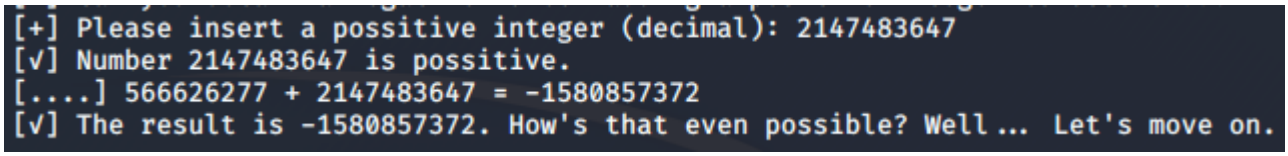

Figure 2.3: Solving the first level.

2.1 Level 1: Basics of addition

2.2 Level 2: Basics of subtraction

2.3 Level 3: Simple equations

2.4 Level 4: Twisting the numbers

[1] https://kde.org/applications/utilities/org.kde.kcalc

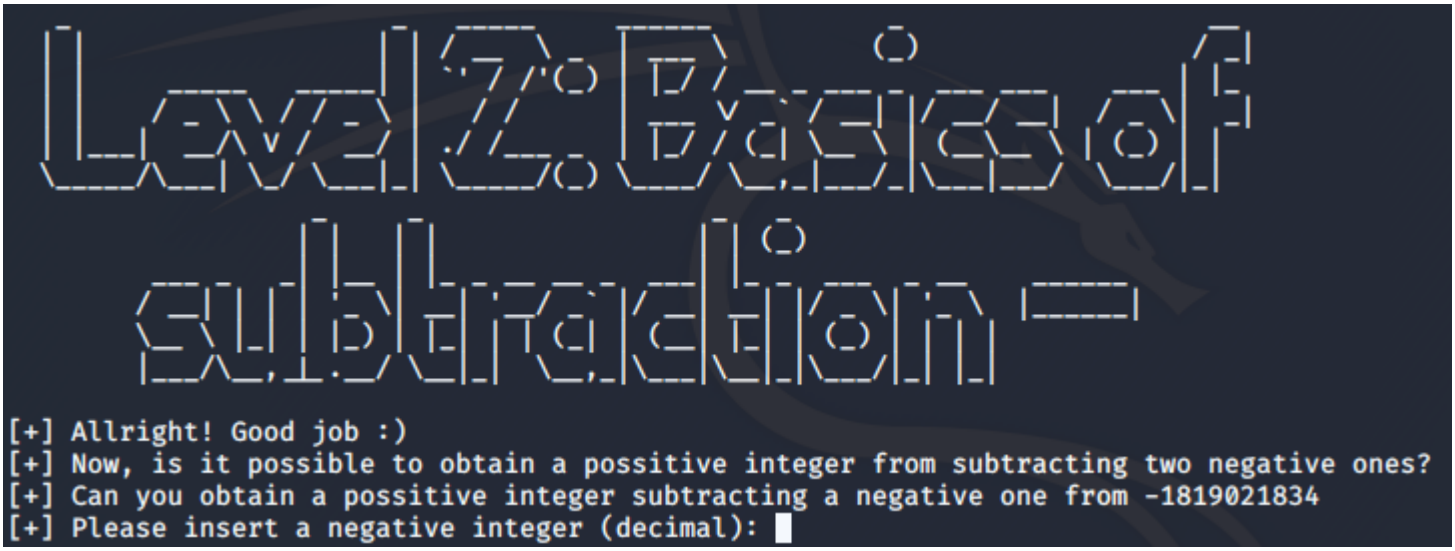

Figure 2.4: Formulation of the second level.

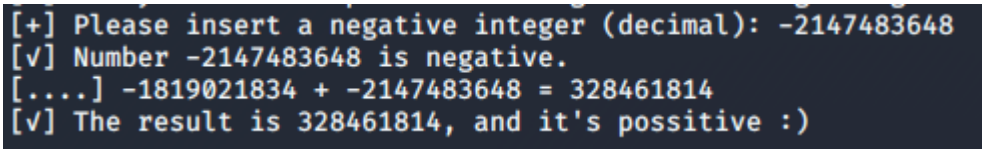

Figure 2.5: Solving the second level.

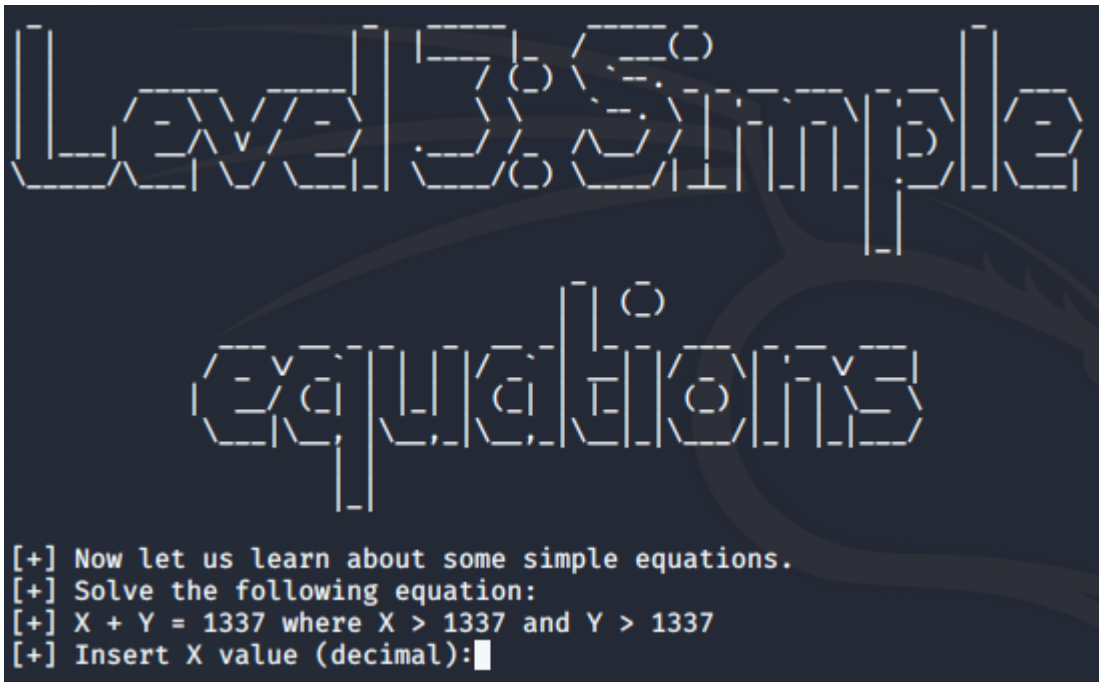

Figure 2.6: Formulation of the third level.

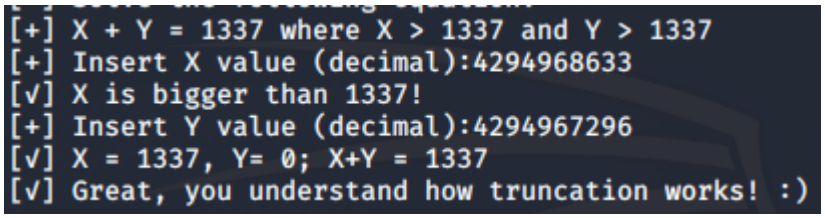

Figure 2.7: Solving the third level.

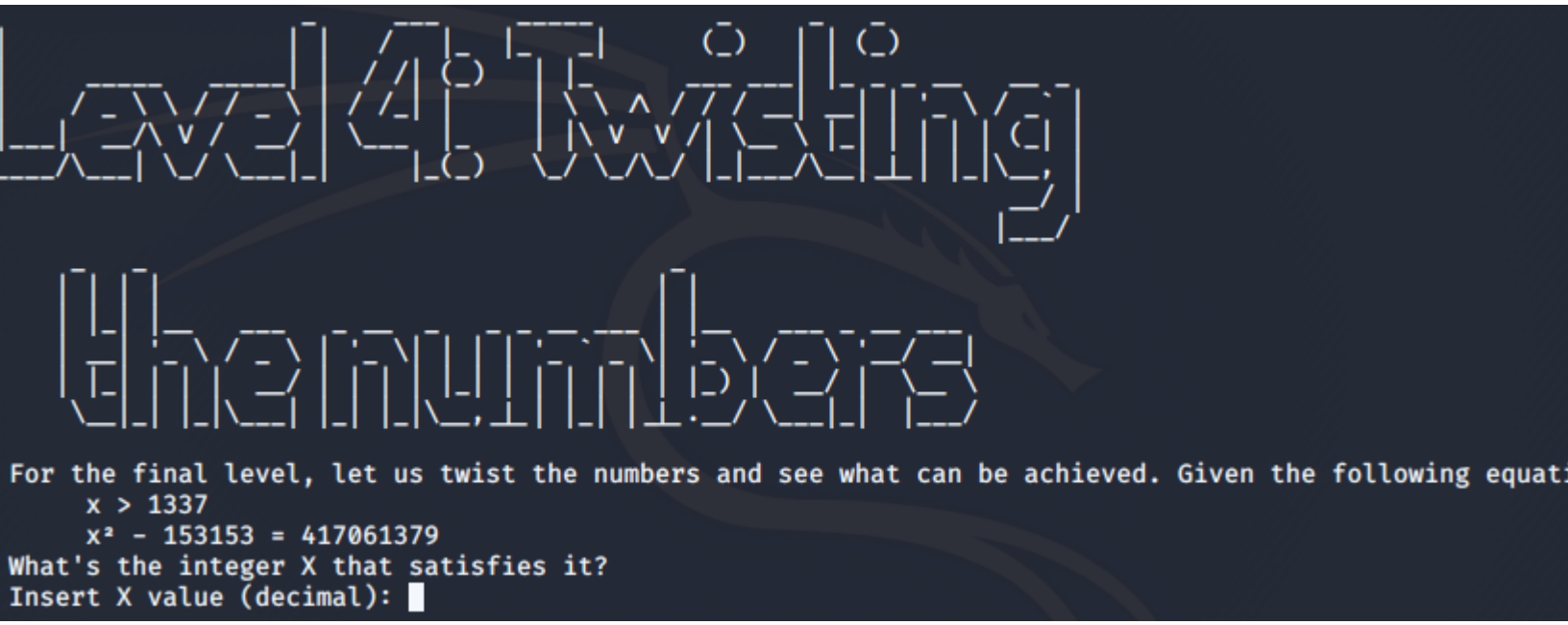

Figure 2.8: Formulation of the forth and last level.

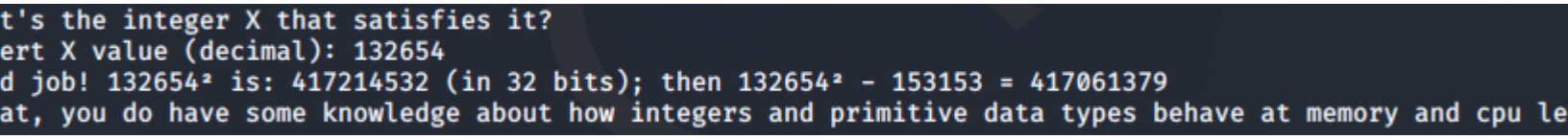

Figure 2.9: Solving the forth and last level.

Acknowledgments

A huge thank Ricardo J. Rodriguez, who first taught me about binary exploitation and inspired me to keep learning and go deeper into it.

Bibliography

[1] Blexim, "Basic Integer Overflows .:: Phrack Magazine ::." dec 2002. [Online]. Available: http://phrack.org/issues/60/10.html (Accessed 2020-02-20).

[2] W. Dietz, P. Li, J. Regehr, and V. Adve, "Understanding integer overflow in c/c++," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 1, pp. 1–29, 2015.

[3] "Integer Overflow - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/computer-science/integer-overflow (Accessed 2020-02-20).

[4] "Integer overflow - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Integer_overflow (Accessed 2020-02-20).

[5] "CWE - 2019 CWE Top 25 Most Dangerous Software Errors." [Online]. Available: https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html#cwe_top_25 (Accessed 2020-02-22).