Bachelor Thesis

March 30, 2025

Understanding arbitrary code execution

A case study on Pokémon Emerald

Sasha Toscano

Abstract

Abstract goes here ... You may include up to six keywords or phrases. Keywords should be separated with semicolons.
Keywords:

Advisor:
Carlo Alberto Furia
Co-advisor:
Marc Langheinrich

Approved by the advisor on Date:

Contents

1 Introduction

2 Arbitrary code execution

I'd like to start with a simple explanation of what arbitrary code execution (ACE in short) is: *updog*. And what is updog, you may ask? *Not much, what's up with you?*.

This is an extremely simple but direct way of explaining what ACE is (credits to Smoovers for the joke). In this case the joke is self-explanatory, but the idea behind it is that we end up saying something (*what's up dawg?*) that is not what we intended to say (*what does updog mean?*). This is extremely similar to what happens during ACE in the real world (don't like this, gotta rephrase), where we end up executing code that we did not intend to execute, albeit with dramatically more complicated consequences.

To get a bit more into the proper definition of things: ACE is a type of flaw or vulnerability that allows an attacker to execute some (generally speaking) malicious code on a target system. This can happen because a machine, by itself, is not capable of differentiating between some generic text and actual commands. Therefore, without proper protections and preventions put in place, an attacker can easily exploit these to gain access to sensitive information, take control of the system, or even cause damage to the system itself. The severity of this incidents can vary greatly: in the past these exploits have gone from allowing gamers to better their speedrunning performances (where speedrunning is the act of playing a video game with the goal of completing it as fast as possible), to leaking sensitive kernel memory on real world systems, as was the case in Retbleed.

These vulnerabilities can be exploited through various means, including buffer overflows, code injection, and other techniques; however in the context of this case study, we will be looking at a specific example of ACE in the world of video games, specifically in the game *Pokémon Emerald (2004)* where the techniques used to exploit the vulnerabilities are **Arbitrary memory write** and **out-of-bounds writes**, which is where existing mechanics are used to place crafted instructions into writable areas.

3 Pokémon Emerald - case study

3.1 The Pomeg glitch

In Pokémon Emerald, the player can use a glitch called *Pomeg glitch* to put the game in an impossible state. To explain this glitch first it would be wise to understand how the game roughly works. In Pokémon Emerald the player can be in two states: the exploration state, where they can go around the map with a team of up to six Pokémon which can be caught in, and the battle state, that is where the fighting happen. Any pokemon caught after the sixth gets sent in the PC which is a storage system that allows the player to store Pokémon that they do not want to carry with them. The Pokémon have their own stats, which improve based on their levels and what other Pokémon they fight. The stat involved with the Pomeg glitch is *HP* (Hit Points), which is the amount of health a Pokémon has. If a Pokémon's HP reaches 0, it faints and cannot be used until it is revived and if all of them reach 0, we get a game over and this can only happen when the game is in the battle state.

The HP stat is calculated through the following formula:

$$\text{HP} = \left(\frac{(2 \times \text{Base} + \text{IV} + \left(\frac{\text{EV}}{4}\right)) \times \text{Level}}{100} \right) + \text{Level} + 10$$

This formula gives us the maximum HP of a Pokémon, which is the maximum amount of health it can have. In the case of the *pomeg glitch*, the only relevant variable is the EV value: this is because the EV value is the only stat that can be directly modified based on the actions of the player. It is a number between 0 and 255, and it increases every time a Pokémon defeats another, but most importantly it can also be decreased by using certain items. In the case of the *pomeg glitch*, the player can use a specific item called *Pomeg Berry* to lower the EV value of a Pokémon by 10. However, if for example the Pokémon's current HP was at 1, when updating current and max HP, the game could set the HP value to 0, or, even worse, something below it, like $(2^16) - 1$ or 65535 HP (due to HP being an unsigned two-bytes integer) and if this was the only pokemon alive, this then becomes a problem because since using this item requires to be in the exploration state, the game has no way to game over.

4 Approach

4.1 The main idea

Some of the various techniques are listed below:

- tech 1
- tech 2

You may write the formulas as follows:

$$\phi(n) = (p-1) \cdot (q-1) \tag{1}$$

To insert a figure use the following command:

Figure 1. The caption of my figure

5 Evaluation

5.1 Results

The experimental result goes here ... [1].

[1] https://www.usi.ch

6 Future work

7 Summary

Future works goes here.