1. Introduction

Imagine you've found a vulnerability in a product. What do you do with this knowledge?

Maybe nothing. There are many situations in which it's perfectly reasonable to decide to go on about your business and let somebody else deal with it. You're busy. You have more important things to do. It's not your code, so it's not your problem. Or maybe it is your code, but you wrote it a long time ago, and it will take a lot of effort to even try to fix it. Maybe the product has already reached end-of-life. Or perhaps fixing this bug will delay the launch of your new product that will supersede this version anyway. There are plenty of good reasons that it might not be worth the hassle of fixing it or reporting it. Although this is what a mathematician would refer to as a degenerate case of the disclosure process in which no disclosure occurs, it's important to recognize that even starting the process of disclosure is a choice one must consider carefully.

Often, though, you will likely feel a need to take some action in response to this newfound knowledge of a vulnerability. If it's your product, you might immediately set off to understand the root cause of the problem and fix it. Once fixed, you probably will want to draw attention to the fix so your product's users can protect themselves. Or perhaps the product is other people's responsibility to fix, and you want to inform them of the existence of this vulnerability.

This is where the process of Coordinated Vulnerability Disclosure (CVD) begins.

< Executive Summary | 1.1. Coordinated Vulnerability Disclosure is a Process, Not an Event >