# Chapter 35
# FPVA: The Manager's View

*Revision 1.0, September 2025.*

## Objectives

- Understand the vulnerability assessment process from the manager's point of view.
- Understand the reasons why organizations avoid such assessments.

## 35.1 Introduction

As a manager, the assessment process can be intimidating or scary. As a result, there are lots of reasons that you might say "no" to such an activity. These reasons can block you from a valuable and productive activity. So, let's look at some of these reasons and how you might respond to them

## 35.2 "Just Say No"

Nancy Reagan was famous for her "just say no" to drugs campaign. Many software managers and developers can have the same response to having their code assessment. The assessment process can be intimidating or scary. So, there are lots of reasons that you might say "no" to such an activity. These reasons can block you from a valuable and productive activity. So, let's look at some of these reasons and how you might respond to them.

### 35.2.1 We Use Best Practices in Design and Development

Your software teams might tell you that they are using the latest in best practices for secure software design and implementation. They may even be very good at testing their own code. However, as the saying goes:

> There's many a slip 'twixt the cup and lip.
> (old English proverb based on Erasmus)

Programmers are only human, so can make mistakes. In such a detail-oriented activity, it is easy to miss something. That's where the assessment team comes in.

It is difficult to have a perfect system without formal specification and verification. And that's something that is quite expensive and rarely done, perhaps only in the most mission critical systems (if even there).

But maybe your teams have done a spectacular job in designing and building their software modules. When you bring together different modules, libraries, or services, the combination of these parts, and the assumptions that they each make, can lead to new and unforeseen vulnerabilities.

### 35.2.2 It is To Expensive

A common and legitimate complaint about an in-depth vulnerability assessment is that it is expensive. And the answer to that complaint is: yes, it definitely can be expensive. However, the cost of not doing it might be much more expensive. Of course, the best time to fix a vulnerability is early in the software development lifecycle. Problems found at the design or coding stage will be cheaper to fix than those found at the testing stage (of which FPVA can be considered a part). Problems found after release and deployment can have significantly higher costs to fix.

| | |
|---|---|
| The best defense is a good offense.<br>*(old sports adage)* | The only real defense is active defense.<br>*(Mao Zedong)* |

As we discussed in one of our earlier chapters, the cost of not doing something can be loss of money, reputation, data, personal identifying information, intellectual property rights, and availability of your systems.

Unfortunately, if you are successful in protecting your systems, you only see the cost of implementing the security process. It can be difficult to tell if you were safe because you were diligent, lucky, or just overly paranoid.

Pretend that you are the CISO of a large company and you are concerned about the software development practices of your company as they relate to security. Somehow, you convince the other leadership that these security issues are important and get them commit 5% of the overall budget to improving these practices. At the end of the year, you are pleased to see that no major security breaches have occurred. However, what the Board of Directors and the investors see is that your company had 5% less profit. It is difficult to argue that the lack of security breaches was due to your increased diligence and not to other factors. This is one of the inherent challenges of being a security practitioner.

We do assessments for the same reason that we buy insurance: the cost of not doing so can be extreme. Finding a vulnerability during the development and testing process is enormously cheaper than dealing with the impact in a deployed system.

### 35.2.3 We Already Ran Static Analysis or Dependency Analysis Tools

Of course, your team should be using static analysis tools (Chapter 36) and dependency analysis tools (Chapter 37) as part of their development cycle. These tools are easy to use, essential, and increase your baseline level of code security and correctness.

However, such tools are no substitute for an in-depth assessment of your code. If your code controls sensitive data such as PII (personal identifying information) or PMI (personal medical information), financial assets, or controls physical devices, you probably need to go the extra step.

> The era of procrastination, of half-measures, of soothing and baffling expedients, of delays is coming to its close. In its place we are entering a period of consequences.
>
> *(Winston Churchill, August 1941)*

The static analysis and dependency tools that are currently available can be pretty good, but they will miss some important classes of problems and can produce noisy results. Our early study compared the results of a skilled analyst to that of automated tools, and showed that tools can miss many critical problems (false negatives) and report noisy, even overly noisy results (false positives)[1]. In some cases, the number of false positives can be so large as to overwhelm the program.

As a side note, the kind of study we performed comparing the efficacy of scanning tools cannot be performed these days on commercial tools because the tool vendors include clauses in their licenses restricting such reports. Such clauses are called "Dewitt Clauses"[2] and resulting in a dangerous and sad state of affairs.

---

[1] J.A. Kupsch and B.P. Miller, "Manual vs. Automated Vulnerability Assessment: A Case Study", *First International Workshop on Managing Insider Security Threats,* West Lafayette, IN, June 2009.
`https://pages.cs.wisc.edu/~kupsch/va/ManVsAutoVulnAssessment.pdf`

[2] A DeWitt Clause forbids the publication of database benchmarks that the database vendor has not sanctioned. The original DeWitt Clause was established by Oracle at the behest of Larry Ellison. Ellison was displeased with a benchmarking study of the performance of various database systems done by David DeWitt in 1982, then an assistant professor, using his Wisconsin Benchmark program, which showed that Oracle's system had poor performance.

### 35.2.4 Reporting Vulnerabilities Will Make Us Look Incompetent

A big concern among software teams and vendors is that they will look bad if vulnerabilities are found in their code.

> A life spent making mistakes is not only more honorable, but more useful than a life spent doing nothing.
>
> *(George Bernard Shaw)*

In the process of performing in-depth software assessments over many years, we have found it is just the opposite situation. We know that all software has flaws. If your software vendor is not reporting vulnerabilities in their code, then either they are not looking (which is definitely bad) or not telling you about them (which is also very bad).

Our experiences have shown that users, customers, investors, and funding agencies have more confidence in you and your software if you have a visible security program that is actively reporting security issues and their remediations.

We have seen this behavior since the first days that we started doing software assessments. In each case, the organization benefited from the finding of vulnerabilities and actively reporting them to their users. Their reputations as a security-aware and security-capable organization thrived.

## 35.3 Summary

- Learned about the vulnerability assessment process from the manager's point of view.
- Learned about the reasons why organizations avoid such assessments.

## 35.4 Exercises

1. With a partner, have a conversation where one of you is an analyst who represents the assessment team and the other represents the software development team. The goal of the analyst is to convince the reluctant developer that such an assessment is important. The developer will come up with as many excuses as possible to not do an assessment, while the analyst tries to motivate them to go ahead with it.

2. Choose a major software or online services company and investigate what are their reporting practices for vulnerabilities:
    a. How promptly does the company report vulnerabilities?
    b. How much detail do they provide in their reports?
    c. Do they report to a central authority such as the National Vulnerability Database (NVD)?
    d. Do they offer bounties for bugs found by outside analysts?
3. Find a case in the news where an outside analyst found a vulnerability in a company's software and the company refused to acknowledge or publish it.
    a. How long after the vulnerability was found did the analysts publicize it.
    b. Did the company ultimately acknowledge the vulnerability?
    c. Did the company ultimately provide any details about the remediation used for the vulnerability?