**Application Security Program: Information Noise**

**TL;DR:** How to not get lost in the cacophony of over-communication and the importance of building trust.

### .....

In my [post](https://www.linkedin.com/pulse/application-security-program-simple-easy-stefano-parini/) about the service aspects of a security program, I introduced the actors of our story: the development team, the business team, and of course the security team. Then I drafted the actor roles as: suppliers and consumers.

The security team, as supplier, will be implementing, managing, and maintaining a service that will rely (among other factors) on the adoption rate to measure its success.

The adoption has two dimensions: breadth and depth. The first is represented by the coverage (itself bi-dimensional how many programs and for each program how many components), and the second is identified with frequency of use. These dimensions are directly affected by the consumer's level of trust.

The development will not trust a service that delivers false information, for a very simple reason: proving that the issue is not an issue takes time, skills, and effort that many developer teams don't have. If you are the developer receiving a report stating that your application is affected by a security issue, first you will verify the finding, then you will have to go back to the security guy.

The business would not trust a service that provides unreliable information. If the service is not accurate, there is an actual risk of spending more for very little reward (in terms of security risk mitigation). Think about a project team spending time to fix something which has very little security impact: if that’s a waste of money per-se, how about the delayed push to production?

Reducing the information noise is a process in and of itself, because the sources of noise are continuously changing. The security policies change over time to adapt to new threats, and the same happens to the security tools.

For example, to ease the triage burden placed on developers, we could suppress the reporting of irrelevant warnings, legacy defects that cannot be fixed, and defects in third-party libraries that are not used.

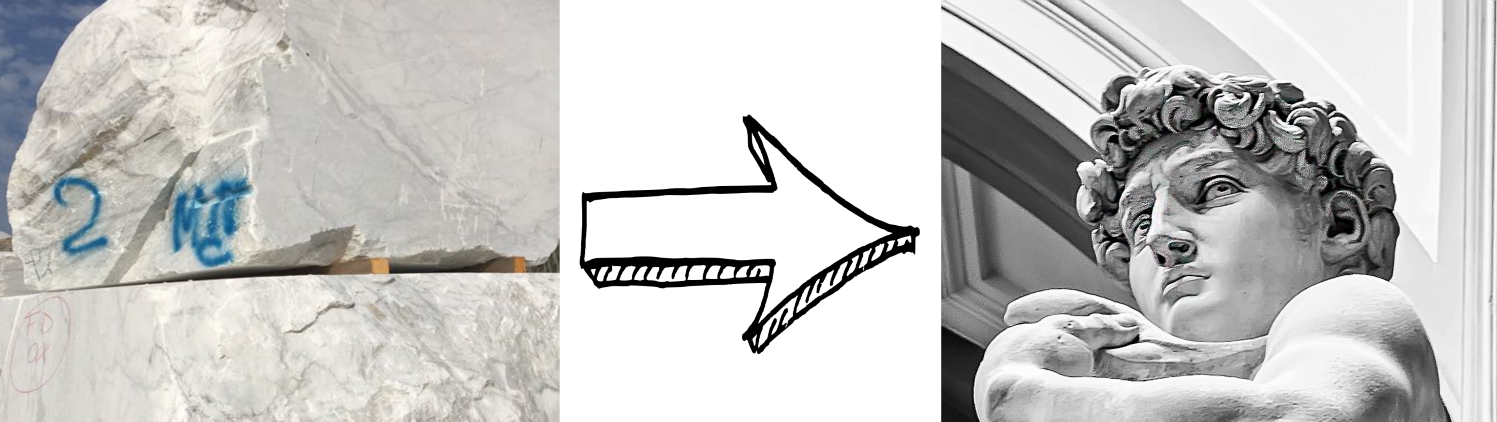
The noise reduction is not a onetime activity and must be treated as an ongoing process that needs to me be continually measured and improved.

### How do you want to do this?

Even if it is counter-intuitive, particularly if you are trying to plugin Appsec in a pre-existing SDLC, at the beginning of the adoption process the focus will be on minimizing false positives at the cost of more false negatives (booom), then tuning the process to make it more accurate.

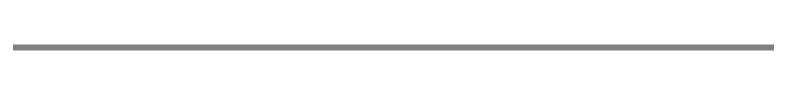
This will also have a positive impact on the issues with VOLUME & VELOCITY (which will be the content of a future post).

Yes! By following these recommendations, you will introduce more false negatives, at the beginning, of course you will, but it will be a refinement process.



For example, to address this problem, the initial identification of these false negatives can be delegated to classic penetration test activities and fed back into the process as corrective actions, so that in the future the Appsec Service will be able to correctly intercept them and filter them out.

Think of it as an MVP approach: probably the first iteration will look nothing like the last one (which is not the FINAL... just the last one).



Disclaimer: the views and opinion present on this post are those of the author and do not necessarily reflect the official policy or position of any other agency, organization, employer or company.