seemplicity

Time to Rethink Over Prioritization & Under Remediation

Enterprises face new security threats daily. Whilst keeping up with the influx of these threats has become a top priority for security teams, the average organization still finds itself with a backlog of more than 57,000 unfixed security issues within 6 months.

To deal with this huge influx of findings, security teams often heavily rely on prioritization. Sounds logical right? If we compare this to our daily lives, in times when we have an abundance of errands that need to get done, we write a to-do list and prioritize what needs to be done first.

The difference with security findings is that the people who are responsible for writing the to-do list are not the same people who are responsible for executing the errands, and the ratio between the "planners" and "doers" is far from equal. To put this into security context, the security team plans the remediation to-do list and prioritizes the most important items to drive forwards, while the development teams actually remediate the prioritized items. What adds complexity is the fact that the ratio of security personnel to developers is normally 1:100, and development teams are often distributed both geographically and organizationally.

As a result of this entire situation, the biggest irony of it all happens — the security team becomes the bottleneck to driving remediation forwards. The scope of remediation can only be as wide as the security team's ability to prioritize findings, while development teams are only aware of items that have been filtered from the to-do list. Even if development teams have the capacity to fix more than has been allocated to them by the security team, they have no visibility into the entire "to-do" list. To sum a long story short, the over reliance on prioritization essentially results in under-remediation.

But it doesn't have to be that way. What if each development team was given the entire to-do-list that's relevant to them, without security acting as the middleman?

You are probably saying to yourself, well without any filtering by the security team, a lot of junk would end up being passed on. You are right, but prioritization doesn't solve that either. What's needed is for the data to be "cleaned up" and then to be properly distributed to the right team at the right time.

With a self-service approach, the scope of remediation can be as wide as the development teams' ability to handle fixes and security teams no longer have to act as project managers. Prioritization then becomes the exception, only in cases where a single development team has more fixes than it can handle. With prioritization as an exception rather than the rule, security teams can be left to focus on real strategic security rather than on project managing tasks.

With a self-service approach the security to-do list is no longer solely owned by the security team, but the ownership is now shared across security and development teams. To put it into "mathematical" terms...

With a Prioritization Approach:

Time-to-remediation

=

Time taken to process by the security team

+

Time taken to remediate by the development team

With a Self-Service Approach:

Time-to-remediation

=

Time taken to remediate by the development team

Learn More at seemplicity.io