Mend.io

A Practical Guide to Making the Most of your SAST Investment

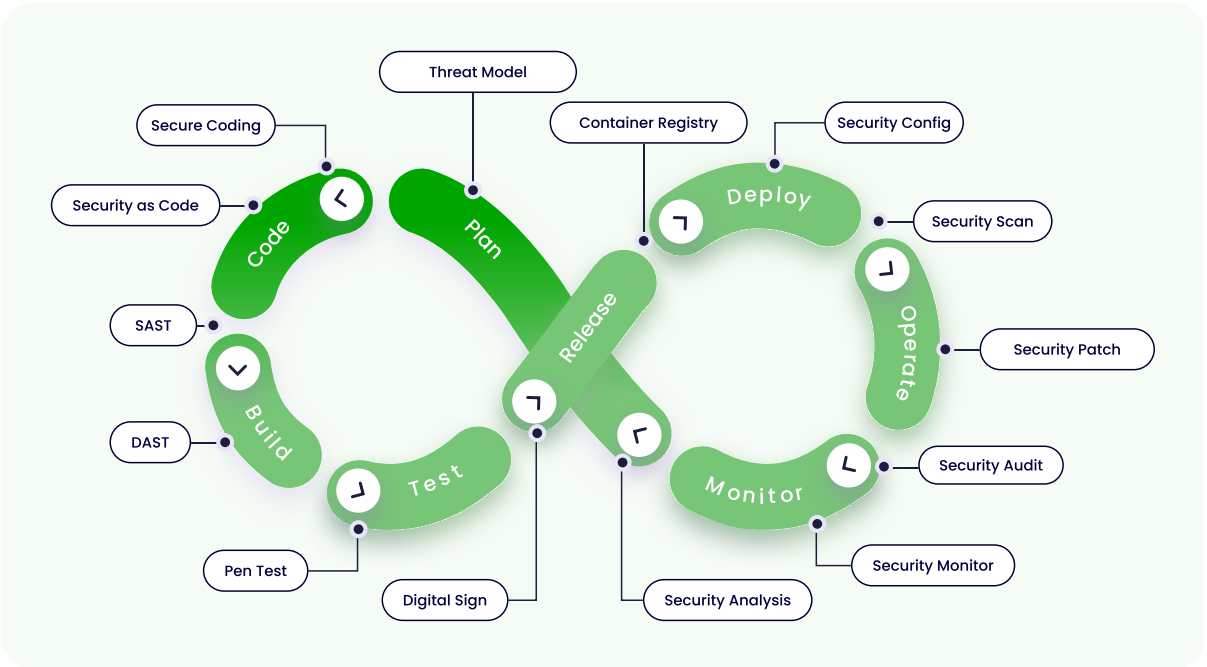
Introduction

We won't sugarcoat it: of all the AppSec tools, Static Application Security Testing (SAST) tools might have one of the worst reputations. And that's a bit of a shame, because done right, SAST tools can deliver real value.

SAST tools use defined rules to locate possible code flaws and send developers alerts flaws that may lead to security vulnerabilities. SAST is, in essence, a code quality tool with a security focus. However, from a developer's point of view, SAST tools tend to identify a high volume of issues that don't pose a real threat. As a result, SAST findings are more likely to be contested by developers—or ignored altogether.

While it's probably not the most expensive tool in your stack, a SAST solution can still be a significant enough investment to evaluate how to maximize your ROI, reach application security goals, and how you can squeeze maximum benefits from your tool. This guide puts together the advice of several experienced security leaders who know what it takes to create an effective AppSec program where security teams and developers both feel productive, respected, and proud of their role in keeping their products secure.

SAST in DevSecOps cycle



5 SAST Purchasing Tips

This is not meant to be an exhaustive buyer's guide, so we won't cover every aspect of the SAST purchasing process. That said, the following five tactics have been recommended by several application security professionals.

1 Effective Dependency Updates

Much of the success of application security programs relies on the buy-in, trust, and respect of developers. They will be the ones tasked with dealing with the tickets that arise from the solution you choose, so they should be brought in at both purchase and implementation time. You don't have to call every single developer to evaluate alongside you; just grab enough that you've covered every team who will be working with you and your security tools.

2 Use Your Own Code For The Proof Of Concept

Vendors will already have demos where they've pitted their product against purposefully buggy applications such as WebGoat. Naturally, vendors want to make their tool look its very best and to helpfully demonstrate what their UI and findings look like. That's great, but you want to see what the findings look like on your code. When they ask you for an application to scan as a proof of concept (PoC), pick one of your own applications, preferably one that has some known quality issues, such as an older version of an application that's already been fixed up. That way, you know what the tool ought to be able to find. Be careful submitting your high-priority application that has been carefully written, as this may result in few or no findings simply because there's nothing wrong with it. If that happens, you can give yourself a nice pat on the back, but you won't have much data to judge the vendor's tool.

3 Ask Thorough Questions About Language Support

Just because a vendor says that their solution covers a language you use doesn't mean it fully covers that language in all types of findings. Ask for lists of which type of code flaws can be found for each language that you currently use, as well as for languages you believe you will be using in the future. Making sure that the solution truly supports the languages you use is another good reason to use your own applications for the PoC.

4 Think Ahead About Your SAST Usage Plans

Will you be scanning your entire code base once a month as part of a compliance process but won't be using the results to address security and code quality concerns? Then it probably won't matter if the tool takes half a day to complete the scan—but you may want to know how easily it can be scheduled. If instead you intend to use your SAST solution more frequently, then metrics like scan time absolutely matter.

5 Communicate Tangible ROI

You might fall in love with the perfect SAST tool but still need to get someone with a wallet on board. According to Nemertes Research analyst Jerald Murphy, executives don't want to hear about cost avoidance as a measure of ROI. Likewise, he advises against estimating loss resulting from a potential security vulnerability or breach. Instead, he suggests looking at the actual costs associated with salary and the time or money saved by staff using that tool. For example, consider the investment in a SAST tool. SAST automates the task of scanning code to find and fix vulnerabilities, drastically reducing the time developers spend doing this manually and freeing them up to focus on higher value tasks. Calculate the time and cost savings resulting from that tool, which is both tangible and repeatable. These are the metrics executive leadership will appreciate.

Improving the Return on Your Existing SAST Investment

Whether you're adding a tool for the very first time or just making better use of the one you've already got, use these tips to make your SAST solution work both harder and smarter for you, your developers, and your applications.

Initial implementation

The goal here is to gently get your processes booted up without your big plans crashing spectacularly.

Scan the right stuff

When you're setting up your SAST tool, don't let it scan your open source files. Seems obvious, given that SAST alerts for open source packages aren't actionable, but it's not an uncommon error. If you forget to exclude your open source files, you'll end up with an unrealistically high number of alerts.

Stay calm and scan wide

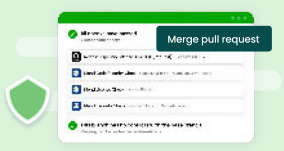
Today is a day that will live in infamy: the day of the first scan, the day of the first gigantic number to set your goals against, the day you scan everything. But the findings are a burden you must carry alone for now. Yes, you will probably see that the number of issues in your applications is quite large and, yes, you may feel a strong urge to address them immediately, but it's best to keep developers out of it at this stage. Demoralizing them early with an overabundance of findings will not start your program off on the right foot. You will get to your backlog but first you must...

Stop the bleeding

Once you've got your baseline number of findings for each application, ease developers into the process of addressing SAST findings by only requiring them to fix those code issues that they're introducing into the code base right now. You do this by making sure SAST results come up with every push to the repo. This gives you and your developers time to do the following:

- Become familiar with the kinds of alerts that SAST will throw at them
- Deal with code issues in a context developers are already very familiar with
- Get training needed to code securely and fix bad habits
- Discover which alerts are not helpful (more on this in the next section)

To be sure, if there's something very obviously bad in the full findings that should simply be fixed right away (passwords hard-coded in plain text, perhaps), go ahead and address that, but otherwise sit on that backlog just a little longer.



Having SAST in your developers' IDEs is a "nice to have". Having SAST set up in your repo for findings to appear after each pull request gives developers feedback right when they're actually looking for it.

Watch your mouth

SAST findings can bring to light developer bad practices, but you need to tread cautiously here. Developers take pride in their work, and while code security isn't unimportant to them, it may not be their top priority or something they strongly relate to. Code quality, on the other hand, is something they absolutely care about, so get on their level. You may get further with developers when you refer to SAST findings in their code as "bugs" rather than "security issues". Don't downplay real and critical issues but, in general, a hot approach just doesn't work; developers have seen too many "the sky is falling!" security alerts turn out to be nothingburgers.

Get QA on board

Ultimately, security is simply a subset of quality so there's no reason for QA to not be involved. If the heads of development teams are letting insecure code through the gate, it's QA who will step up as your next line of defense and stop them. Get QA involved as early into the process as possible, make sure they're aware of your baseline numbers for overall vulnerabilities and critical ones, and set goals with them.

Balancing and fine-tuning

The goal here is to be neither under- nor over-implemented. Care about code quality and security and be helpful, but don't frustrate developers and block them from getting their work done.

Find out what's annoying developers

A SAST scan can produce a discouragingly large number of results. While none of those results are truly "false positives", they are not all equally important or indicative of a meaningful problem in a given environment. Senior developers will gladly tell you which types of alerts don't carry much weight for their particular application if you just ask. Listen to developers, and turn off the kinds of alerts that they don't find helpful. "Heap inspection", for instance, is a common alert type to hit ignore on forever because it isn't a very viable attack vector in most modern environments.

(Mostly) ignore tool-provided severity

Virtually every SAST tool will give each type of finding a severity level in order to help you prioritize your fixes. If you want to be really effective, you won't take these rankings as gospel. SAST severity scores are usually based on a worst-case scenario that doesn't reflect the situation your application is in.

Some types of alerts will point to things that are universally worth fixing, but many are completely dependent on the context of the application. The more types of alerts you can turn off because they don't matter (make sure you know that's true!), the less annoying SAST will be. And since you'll soon be addressing your backlog based on severity within the context of each application, take notes now.

Put a developer on your security team or train your security team in basic development to better interpret SAST findings and their relevance to your applications. Security Champion programs are also effective.

Paying down that debt

The goal here is to be neither under-nor over-implemented. Care about code quality and security and be helpful, but don't frustrate developers and block them from getting their work done.

Set good goals

Before you start working on that backlog, you need to set some goals. Figure out what you're going to track, how you're going to measure it, and how much positive change is reasonable in a given time frame. Make sure to keep both development and QA in the loop and take their feedback on board. Start thinking about what bad indicators will look like and what you'll do if you see them.

Prioritize your applications

OK, ready to address that backlog now? You need a plan. Take an inventory of your applications and prioritize them in a way that makes sense for your organization. Most teams start with customer-facing applications, and prioritize those based on which ones customers can themselves scan and have already expressed concerns about. This isn't a big revelation; we just want to make sure it isn't forgotten.

Take a trickle approach to your backlog

It's finally time to address that backlog of issues. Are you going to tackle every issue all at once? Not if you want to keep developers happy! Starting with your highest prioritized group of applications, begin addressing one kind of code issue at a time. Start with the ones that are the highest level of severity (SQL injections and deserialization of untrusted data are common choices) and have your developers address every alert of that one type. Next month, sprint, or however you're dividing time, have them address the next type of alert. Continue until every alert that you care about is addressed, then start working on the lower priority applications in the same method. Remember to keep your developers aware of the plan so there aren't any surprises.

When you have many developers working on an application at breakneck speeds, it's easy for code quality and security to slip away. SAST exists to help with that exact problem, but when tools are not fine-tuned and their findings aren't delivered with care, they can end up as simply not-so-great investments that get in the way. We hope the advice from this guide helps you get the most from your SAST purchase and set your AppSec program up for success.