VERACODE

SPECIAL REPORT

Communicating Application Security Success to Your Executive Leadership

Content Contributed by the Veracode
Customer Advisory Board Working Group

Contents

INTRODUCTION

This paper, the result of a collaboration between Veracode staff and the Veracode Customer Advisory Board, was created to help security professionals prove the effectiveness of their application security programs.

Veracode's Customer Advisory Board (CAB) is made up of application security professionals in several industries, managing teams of various sizes at differing stages on their AppSec journeys. But despite these differences, when the Board meets to share best practices and lessons learned, one common theme always emerges — change.

Software development, and software security, have changed dramatically in recent years. As the move to the cloud and the shift to DevOps brought architecture changes including serverless functions and microservices, software security has had to adapt. Static analysis (SAST), dynamic analysis (DAST), and software composition analysis (SCA) solutions continue to mature and include checks for new vulnerability types, while new testing types like interactive analysis (IAST) have emerged. Penetration testing remains a reliable, but extremely expensive, technique. But software vulnerabilities remain, and attackers have also matured and improved their capabilities to target the new landscape. In fact, Veracode's most recent *State of Software Security* report found that 76 percent of applications have at least one security flaw on initial scan. And headlines continue to feature high-profile data breaches.

Even in the face of all this change and continued threat, CISOs and application security program owners across all industries are faced with budget constraints and scrutiny.

These constraints often raise the following questions:

1. **How do we determine and justify the required resources for an application security program?**
2. **How do we ensure, and prove, that development teams are adopting software security practices?**
3. **Is our application security operating effectively? And how do we prove that?**

Answering these questions is no small feat, in part because of the ways that application security differs from other security solutions. You don't install an AppSec tool and count the breaches getting deflected; you change the way you develop software by building security in from the start. This is a significant pivot from traditional, reactive ways of thinking about security. Consequently, even after security professionals make the case and secure funding for an AppSec investment, explaining what application security success looks like and proving the effectiveness of their program is not easy.

The members of the Veracode Customer Advisory Board all experienced this challenge in some way. To help each other and their wider set of AppSec peers, a subset of CAB members formed a working group to discuss and ultimately answer the above questions. This paper is the result of the working group's efforts. Produced to capture the group's collective answers, the paper will contribute to a set of industry best practices that help organizations mature their AppSec programs, measure their success, and ultimately lower their risk.

> **"You can't manage what you can't measure."** PETER DRUCKER

The paper defines a set of metrics that CISOs and application security program managers can use to establish, drive adoption, and operationalize an application security program (see Figure 1). These data points can help inform decisions at different stages of program maturity.

It can also answer the basic question often asked by the Executive Team and the Board: is the application security program effective or not?

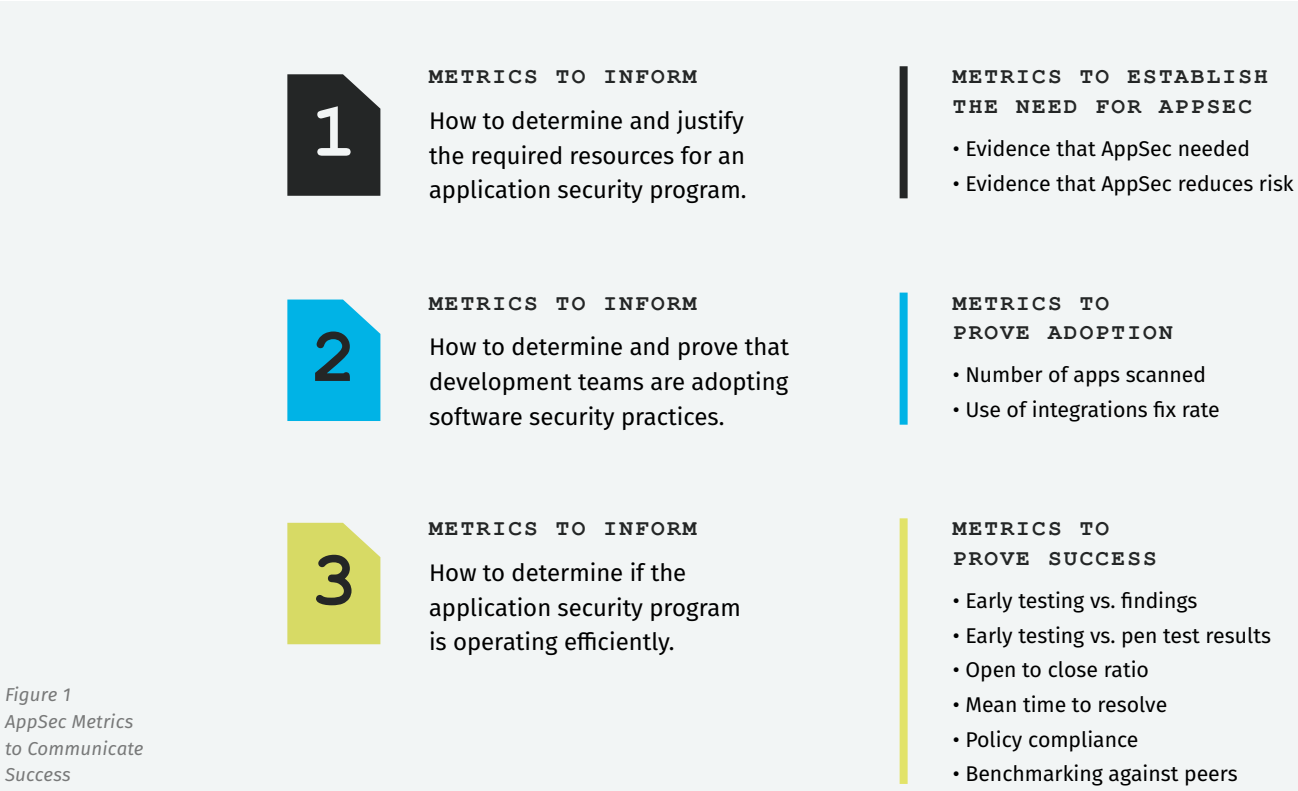


Figure 1
AppSec Metrics to Communicate Success

1 How Do I Prove That I Need an AppSec Program?

AppSec managers need a justifiable AppSec approach and dataset that set parameters around the program, give a starting point, and set up how the program will grow over time. That approach starts with providing evidence that an application security program is necessary and that it will reduce risk.

Why do we need AppSec?

The evidence supporting the need for application security is substantial and growing. As cited above, Veracode's most recent *State of Software Security* (SOSS) report found that 76 percent of applications contain a security flaw on initial scan. And the latest *State of Software Security: Open Source Edition* found that more than 70 percent of apps contain a security flaw in an open source library on initial scan.

These numbers take on new significance in light of the recent acceleration of digital transformation.

↑34%

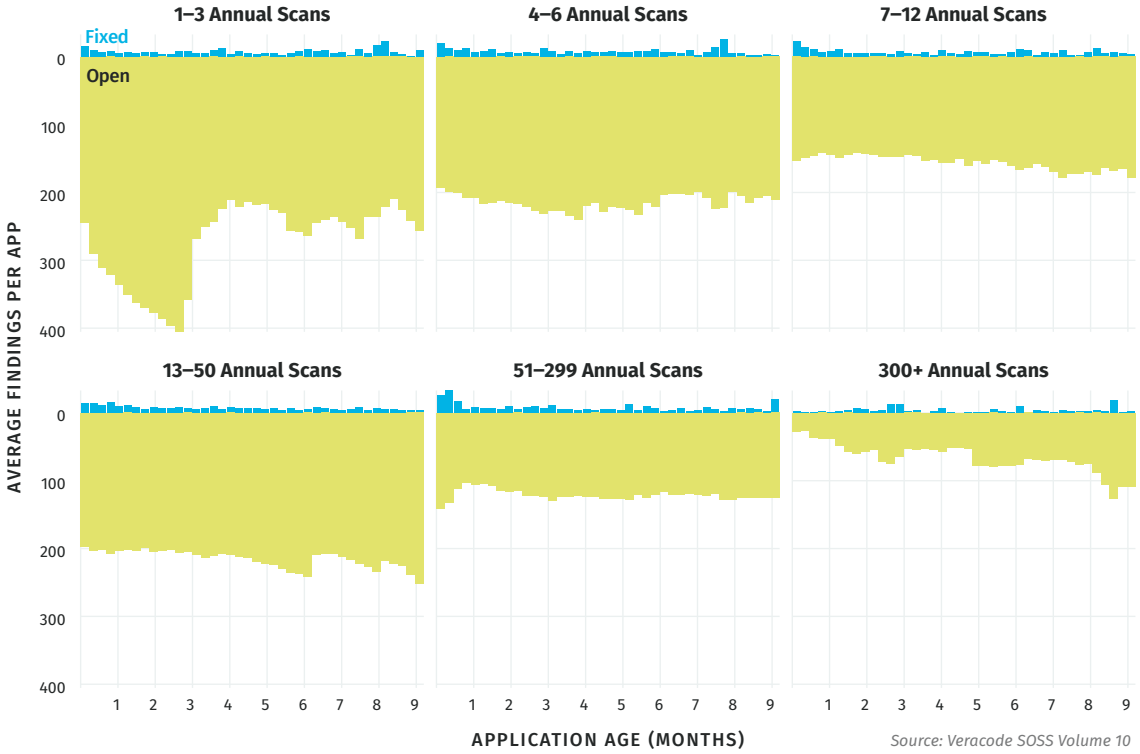
In fact, cyberattacks are up 34 percent since the start of the pandemic.

$3.86 million

And the Ponemon Institute's 2020 *Cost of a Data Breach* report found that the average total cost of a data breach is $3.86 million.

Are AppSec programs effective?

Veracode's *State of Software Security* report also contains some good news — it reveals data on subsets of organizations that are following certain best practices and are dramatically lowering their risk. For example, the 10th volume of the report found that those scanning their apps for security more frequently had significantly less security debt than those scanning the least (see Figure 2).



Source: Veracode SOSS Volume 10

Figure 2
Comparison of fix capacity and security debt by scan frequency

In addition, the 11th volume of Veracode's SOSS report found that those scanning more often fix security flaws faster (see Figure 3).

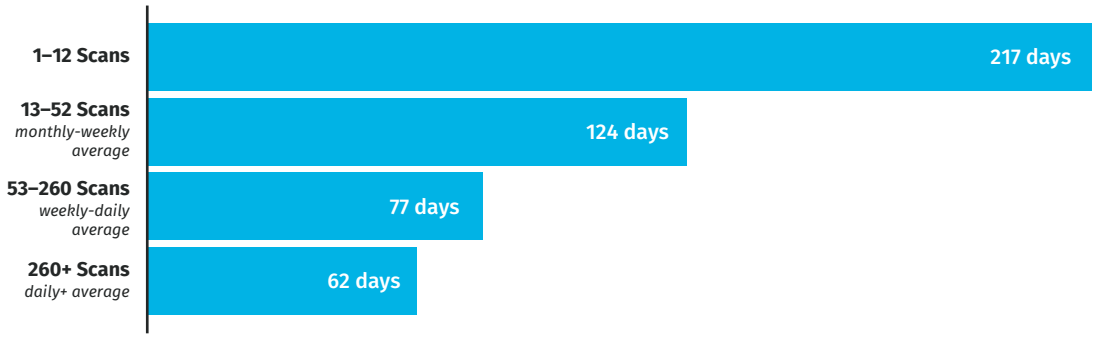


Figure 3
Time to remediate 50% of flaws based on scan frequency

Charts like Figures 2 and 3 are a good way to illustrate what your program should be aiming for and why.

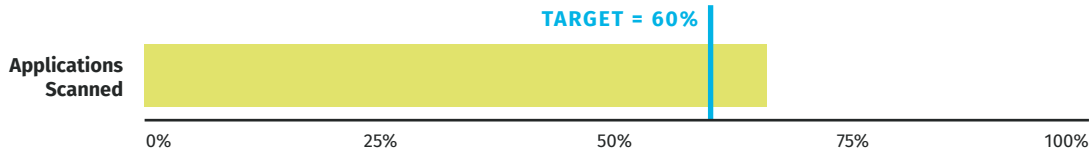
2 How Do I Prove That Developers Are Participating in My AppSec Program?

AppSec success hinges on development buy-in and engagement. Therefore, proving that your AppSec program is effective requires evidence of developer adoption. Metrics, like the following, prove that development teams are adopting software security practices.

Are developers adopting AppSec?

Figure 4 helps illustrate the adoption of an AppSec program, answering the question, "Are development teams scanning the applications they produce?" This is especially helpful in a program's early stages as development teams change their practices to incorporate security testing. Notice the addition of the target as well. Including the goal or target is key to putting your metrics in context when communicating with executives.

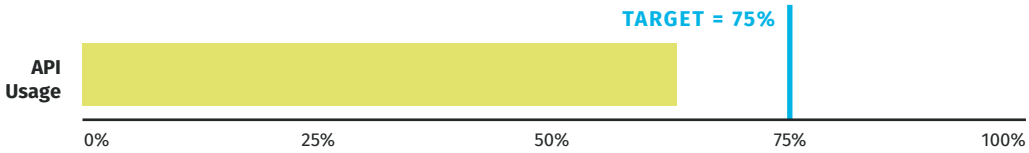
Figure 4
Measure the percentage of apps scanned



Is security testing integrated into developer processes?

It is important for executives to be aware that AppSec success relies on developer participation. And developer participation relies on ease of use and training. In turn, the most effective AppSec programs are invisible to developers — they simply integrate seamlessly into their processes and tools (see "Communicating the Story" below for more on how to advise executives of this concept), and they include developer training on secure coding. A recent ESG report found that one of the biggest challenges to application security is the lack of formal developer training.

Figure 5
Determine whether security is integrated into development processes



An important metric to highlight application security success is the rate at which development teams are taking advantage of APIs to integrate security into their processes (Figure 5). For example, are developers using one of Veracode's APIs or integrations that enable them to scan for security through their IDE or build server, or that automatically send security findings to their bug tracking system?

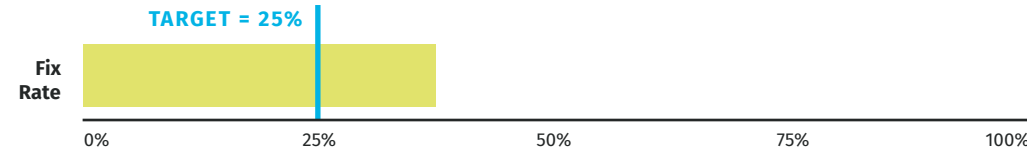
Additional support for this metric again comes from research from the Veracode *State of Software Security* report, which recently found that those organizations that scan via API shorten the time to address half their security flaws by 17.5 days.

Note that developer participation in AppSec also requires an executive mandate and buy-in from the CTO. Establishing buy-in at that level from the outset is critical.

Are developers fixing what they find?

Simply finding software vulnerabilities is not the end goal; fixing flaws is. Figure 6 illustrates total number of findings closed divided by total number of findings that were open. This is especially useful to look at by team or business unit to compare AppSec adoption, as well as by category to help understand where training or resourcing investment is needed.

Figure 6
Find out how effectively teams are resolving security findings



3 How Do I Prove That My AppSec Program Is Making Us More Secure?

There is often a misconception that AppSec is one tool or one project with a finish line, creating confusion about results and outcomes. In reality, it's a process, not a project. Rather than a one-off initiative, effective application security is ultimately a component of the software development process, just like QA, and the measures of success need to reflect that. Use the following metrics both to guide the progress of your program and to illustrate its progress and success to executives.

> "If you can't describe what you are doing as a process, you don't know what you're doing.
>
> W EDWARDS DEMING

Is early testing reducing later findings?

When trying to prove the success of an AppSec program to an executive team, one key metric is the correlation between security activities early in the development process and the number of security flaws found in a release candidate or in production.

This metric supports the hypothesis at the core of most AppSec programs — it is more cost-effective and efficient to reduce risk to your organization by closing security findings early rather than delaying time to production.

For example, Figure 7 shows the relationship between security testing early in the development process, in the individual IDE of a software developer, and the number of flaws found in the release candidate.

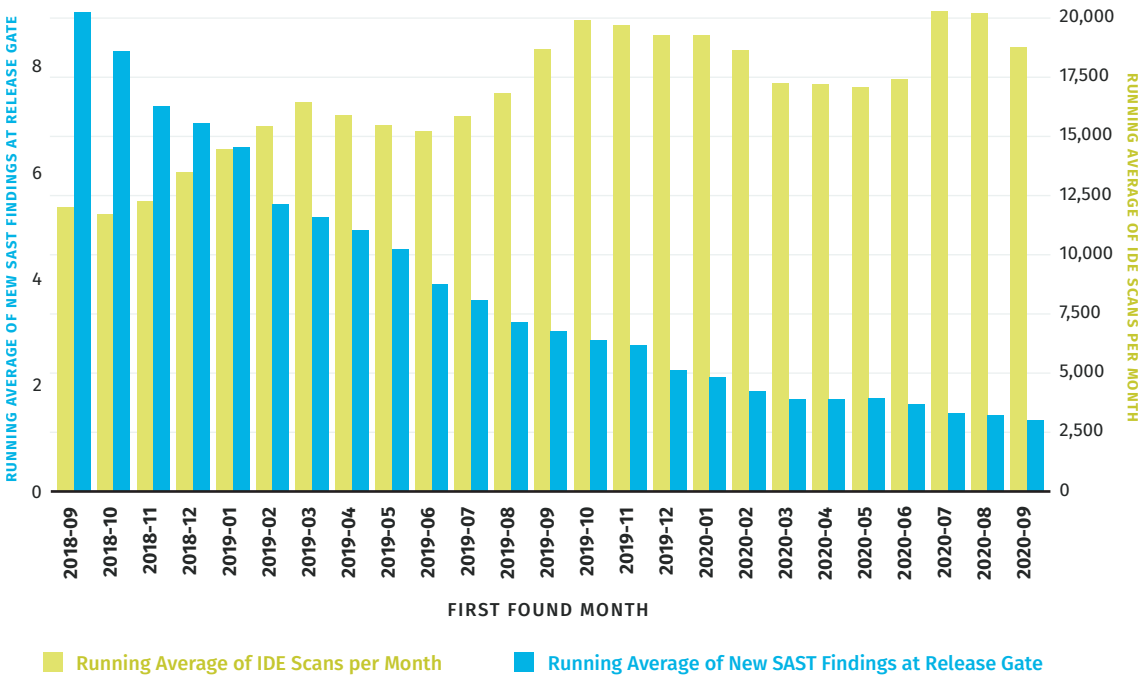


Figure 7
Impact of IDE scans on flaws in release candidate

Figure 7 (based on real Veracode customer data from an organization with 1,200 developers) clearly highlights the success of an application security program by depicting the correlation between scanning for security in the IDE and security flaws found just before production. This customer has incorporated security analysis early in the development pipeline, where it becomes a form of security training for developers, both alerting to and educating on security vulnerabilities — ultimately preventing both immediate and future security flaws. As the IDE scanning ramps up (yellow bars), the flaws in the release candidate go down (blue bars), reducing risk when that candidate hits production.

A developer provided the direct feedback that, "[Performing security testing early in the IDE helps with] learning how to identify and avoid potential issues in the first place. Such as avoiding common, but less obvious highly insecure patterns like SQL Injection and XSS (Cross-Site Scripting)."

Ultimately, this "early testing vs. later flaws" metric is effective because it addresses an area that has been particularly challenging for security professionals — proving that change in the development process is having a positive impact. It illustrates how those changes are having a downstream effect on reducing the costly exercise of fixing security flaws in production.

NOTE

This metric can be a challenge for organizations just starting an application security program. Although the metric above is attainable for any organization of any size, those just starting out will want to kick off their programs with activities that will quickly reduce risk and see positive results. That early start point is typically relying on static analysis as a release gate. In this case, you won't have two metrics to measure and compare. You will only have the one measurement — number of flaws in each release/number of releases blocked. In this instance, it would be useful to look at time as a dimension: are you seeing fewer flaws over time? Is the team fixing more than they are finding? Are releases blocked less often? This is not nearly as compelling, and in these cases, other metrics, such as policy compliance or peer comparison, may be more appropriate.

Are we fixing more security flaws than we find?

Figure 8 illustrates whether the organization is resolving more security findings than it is identifying. If the company is closing more flaws than it is locating, it is truly reducing risk, rather than just adding to a tech debt backlog (as in Figure 9).

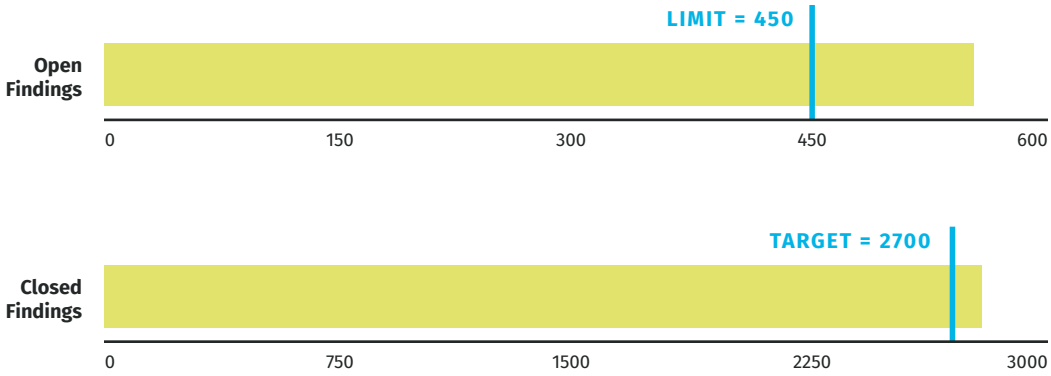


Figure 8
Compare the number of security findings to number of closed security findings

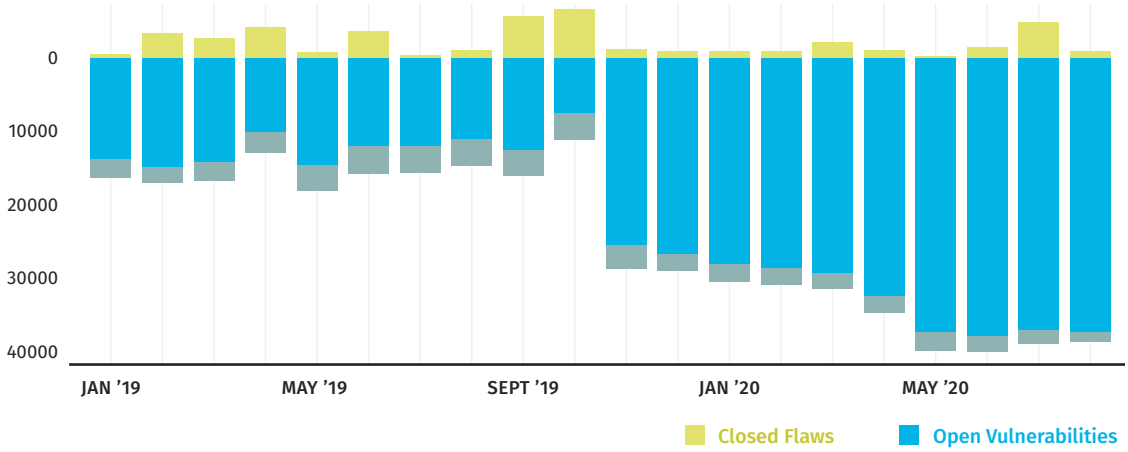


Figure 9
Security debt is the result of opening more flaws than closing

Are we fixing security flaws quickly?

Figure 10 illustrates the average time to address security findings. The standard definition for mean time to resolve (MTTR) is corrective maintenance time/total number of corrective maintenance actions. MTTR is key for motivating teams in the spirit of continuous improvement. Keep in mind that this metric is highly dependent on the context of your organization. If you have an internal-facing legacy system, an average time to resolve for that application of 30 days may be great. If you have an external application that handles your PII, five days may be too long for your average time to resolve.

Figure 10
Mean time to resolve

5 **Average Days to Resolve All Findings**

2 **Average Days to Resolve Policy Findings**

Are our apps passing security policy?

The percent of applications in your AppSec program that are in compliance with your AppSec policy is a clear and concise way to prove AppSec success.

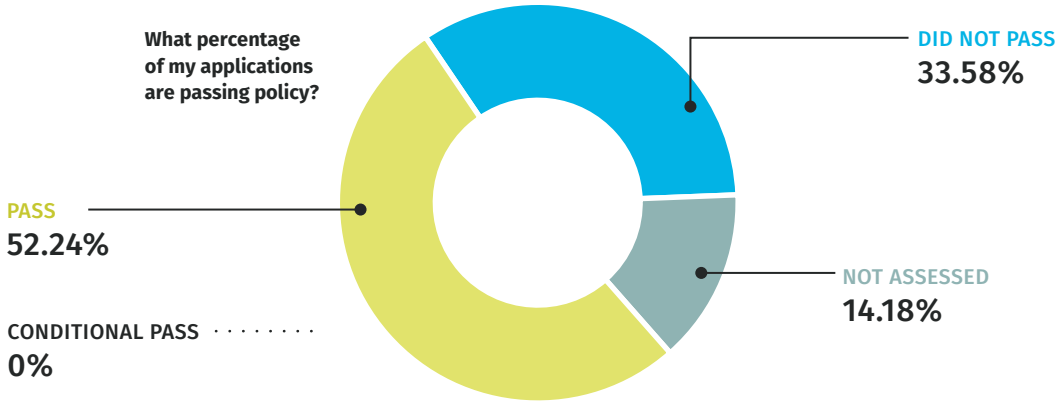


Figure 11
Determine whether your applications are passing policy

However, the success of this metric depends on whether you have established the right policy for your organization. For instance, if security is being introduced or enforced for the first time, begin with some achievable policy standards. Start with a simple policy: no high or very high critical flaws. Then get more stringent over time as developers adopt security into their daily routine. As the security assessments and remediation become part of the development process and developers become more accustomed to remediation, policies that aim to comply with PCI or OWASP requirements become more realistic rather than at the beginning.

In addition, not all apps are created equal, so create different requirements for different apps. For instance, an application that has IP or is public-facing may require all medium to very critical flaws to be fixed. A one-page temporary marketing site may only require high/very high flaws to be fixed.

Be sure to provide executives with context when policy metrics are presented by including targets.

Are we spending less pen testing dollars?

You could look at the relationship between security assessments and security training early in the development cycle and the results of pen testing in production. But keep in mind that pen testing will always find flaws that automated scanning will not, so simply measuring the number of pen testing findings might not be adequate.

As you increase the amount of security testing earlier in development, you should, however, see a difference in the types of flaws found by pen testers. In other words, you won't be wasting expensive pen testing dollars on flaws that automated scans could have picked up. You'll be optimizing your pen testing budget on complicated flaws that truly need a human eye to identify.

While pen testing is an important security check, remember that the results are a point in time only. Automated scanning allows for an iterative approach to security checks continuously throughout the development process.

A Veracode pen tester recently wrote a blog post that speaks to this issue. He points out that he regularly spends time on vulnerabilities that are easily and less expensively identified and remediated in the development phase. Using a pen tester's time to work on these vulnerabilities is much less efficient, and much more expensive. You could illustrate AppSec success by comparing the number of early security tests (as in Figure 7) with the types of flaws identified by pen testers. You should see a difference in types found as the number of tests in the IDE increases.

PEN TESTING

Pen testing is best used to find and exploit business logic or architectural flaws, conduct manual exploitation of vulnerabilities to show business impact (i.e., extract data from a SQL Injection vulnerability), and to provide real-world attacker simulation.

AUTOMATION

Automation is ideal for finding commonly "known" vulnerabilities. For instance, input validation flaws like XSS and SQL Injection can be easily found with automation. But for more complex flaws, such as Insecure Direct Object References (IDOR), automation has a hard time understanding the "logic" behind these types of flaws, which is why manual testing is recommended in addition to automation.

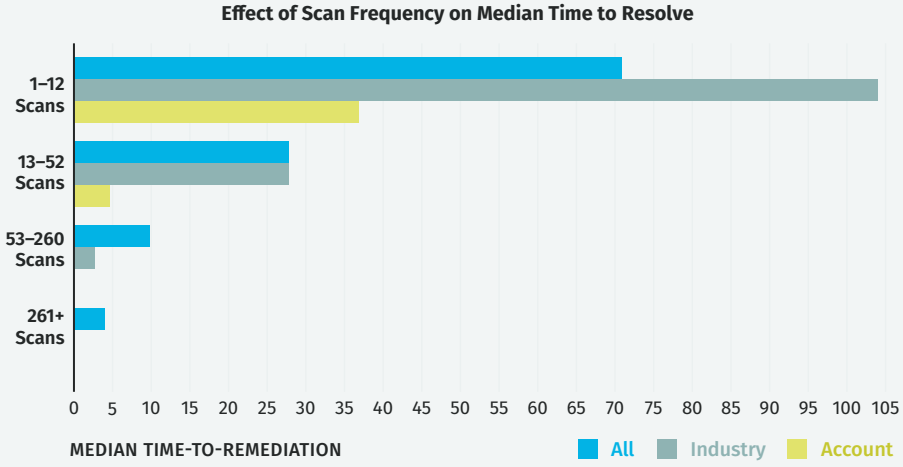
Is our AppSec program as effective as our peers'?

Another powerful metric to consider to communicate AppSec success to executives is peer benchmarking. Provide executives with a frame of reference using the state of software security among peers and the comparison to the current state. For example, Veracode's annual *State of Software Security* report analyzes software security data by industry, and you can compare your numbers to others in your industry and all other Veracode customers. See the chart below as an example. While this customer is scanning only a few times a year, those applications that are scanned more frequently have a much faster time to resolve. So, the recommendation would be to move more applications to a more frequent scanning cadence, and (hopefully) the customer would see an even faster time to resolve — like other customers in their industry and in the market overall.

Does DevSecOps drive faster fixing?

Effect of scan frequency on fix rate and time-to-remediation

In general, we expect a DevOps-oriented team to conduct frequent security scans of their code at regular intervals during the development lifecycle. Furthermore, we'd hope to see evidence that those behaviors correlate with faster fix timelines. This chart to the right provides visibility into where this is true.



How often are applications tested?

Frequency of security scanning across applications

More frequent scanning correlates with a marked improvement in remediation timeframes. With this in mind, the chart to the right provides insight into how frequently applications are tested.

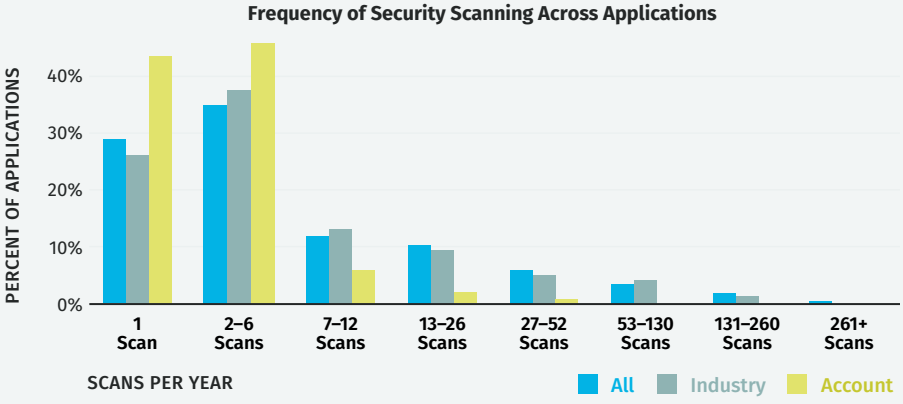


Figure 12
AppSec peer benchmarking

How Do I Frame the Story of Our AppSec Success?

Because of the nature of application security, simply presenting a set of metrics might not be adequate to convey success. You may need to add further context and background to tell the story of your AppSec progress.

FOR EXAMPLE

Defining AppSec vs. other security technologies

Level set on what application security is. For instance, explain that unlike other security solutions, like a WAF, application security is not a tool, but an ongoing program integrated into product development. This program involves changing developer behavior and processes, and risk will not reduce overnight. Over time, security needs to become something that is owned and managed by the development team, just like they own quality testing and scalability. Application security is also more heavily focused on prevention than other security technologies and is therefore often not as easy to quantify. Prevention is always a "tougher sell," in security, and in other areas of life. Take fire safety: Teaching your children not to play with matches is just as important in fighting fires as a fire extinguisher, but it's easy to point to the number of fires you put out with the fire extinguisher. It's less black and white to quantify the increase in safety from educating your children. How many fires did you prevent by explaining the danger of matches and keeping them out of reach?

In terms of application security, it's important to drive home the point that the least expensive security flaw to address is the one that is never created, and the least disruptive vulnerability is the one that is never exploited. You won't see immediate and dramatic numbers of breaches thwarted, but you will see, over time, fewer entry points for cyberattackers in your production code.

Successful application security programs are about more than tools. They are also about more than blocking cyberattacks. Rather, they educate developers on creating secure code and also enable them to test code for security while writing it. In addition, they test code for security vulnerabilities at each stage of the development process, from coding to production. Finally, successful programs incorporate remediation guidance so that the output is not simply identification of flaws, but remediation of them.

Outlining the roadmap for your program, and what the goals are at each stage

Start by defining what good looks like — where should we be headed based on our application landscape and risk tolerance? Then explain which stage of maturity you are in, the different phases of the program that you will go through to reach the end goal, and what the metrics look like at each phase.

FOR EXAMPLE

For example, a company that is just starting out with AppSec will likely need to spend time on the following metrics:

1. **Identifying applications for program inclusion based on risk posture and design**

   How many applications are in the initial scope?

2. **Working with developers to initiate scans and setting up integrations to pull scan results into the appropriate ticketing system to ensure visibility**

   How many applications have been scanned at least once?

   Are the scan results available for the appropriate dev team to action alongside the other bugs in their backlog?

   How many applications have been rescanned?

3. **Setting an achievable security policy and remediate timeline**

   How many findings have been disallowed by policy?

   How many findings are outside of their grace period for remediation?

A company that has invested heavily in AppSec will be likely looking at the following metrics:

1. **Do all new development projects and applications include a security design review and assign the appropriate security policy based on risk posture and design?**

2. **Do all active dev projects have security testing integrated into their pipelines?**

3. **Is a security scan required as part of the release process? Does a failed security scan break the release to production?**

4. **How quickly are developers closing findings that impact policy? Are there trends in certain CWE types or categories that may suggest training improvements?**

CONCLUSION

Answering “what does good look like?” can be challenging in application security.

Even after security professionals make the case and secure funding for an AppSec investment, explaining what application security success looks like and proving the effectiveness of their program is not easy.

The members of the Veracode Customer Advisory Board came together to address this challenge and provide some answers. Using their collective experiences managing a diverse set of application security programs, they established a set of metrics that should help AppSec managers at organizations of any size and in any industry communicate the effectiveness of their programs. The CAB working group ultimately decided that communicating AppSec success relies on metrics that establish that the program is necessary, is being adopted, and is truly reducing risk. In addition, they found that the metrics alone are often not enough, and require the right context in order to communicate AppSec success effectively.

We hope the guidance in this paper helps shape and promote your program and ultimately reduce your organization’s risk.

PLEASE CONTACT US WITH QUESTIONS ABOUT THIS CONTENT OR OUR CUSTOMER ADVISORY BOARD.

VERACODE

Veracode is the leading AppSec partner for creating secure software, reducing the risk of security breach and increasing security and development teams’ productivity. As a result, companies using Veracode can move their business, and the world, forward. With its combination of automation, integrations, process, and speed, Veracode helps companies get accurate and reliable results to focus their efforts on fixing, not just finding, potential vulnerabilities. Veracode serves more than 2,500 customers worldwide across a wide range of industries. The Veracode cloud platform has assessed more than 14 trillion lines of code and helped companies fix more than 46 million security flaws.

www.veracode.com Veracode Blog Twitter