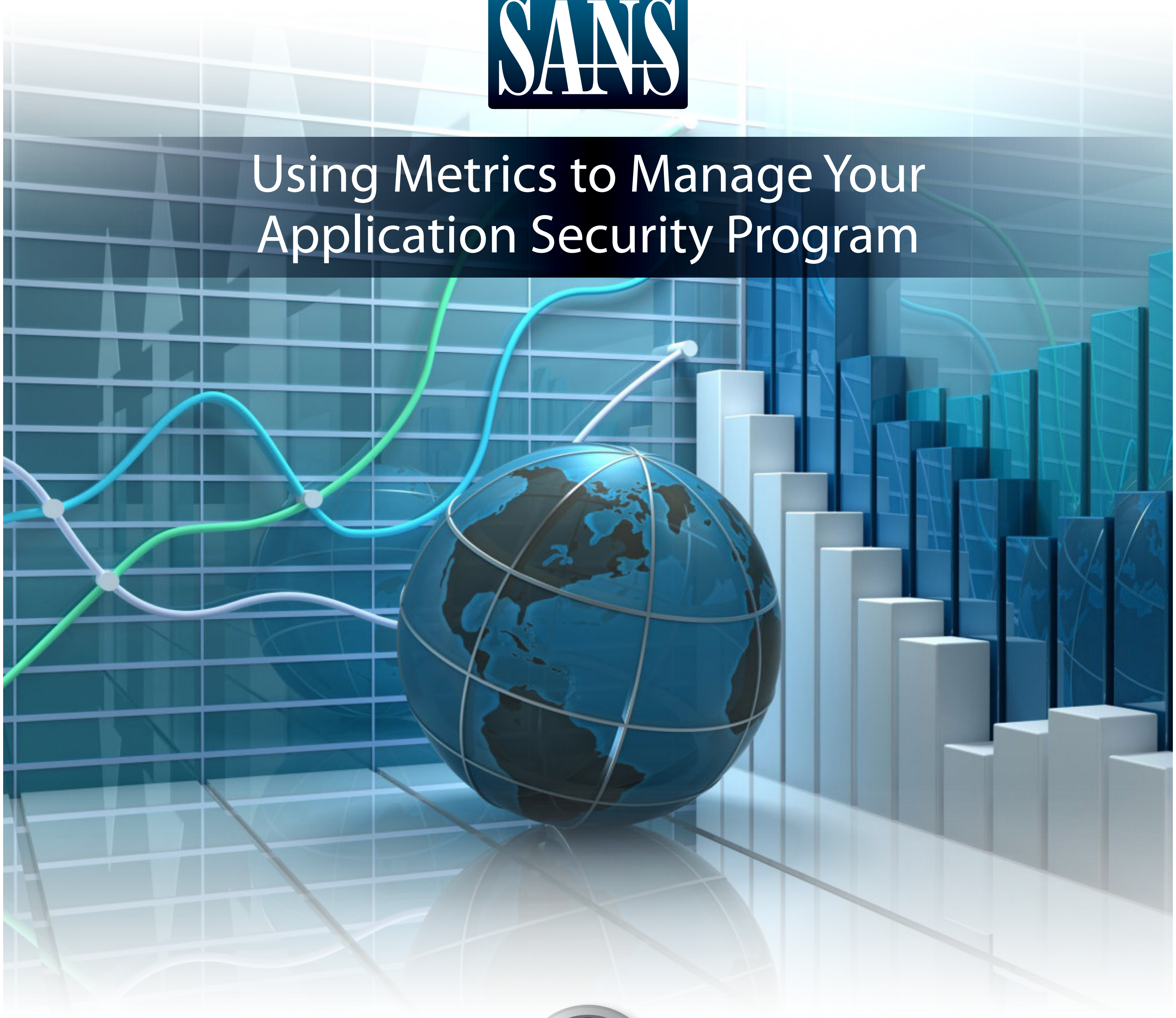
Using Metrics to Manage Your Application Security Program



Written by Jim Bird

March 2016

Sponsored by
Veracode

In this paper, we'll look at the first steps in measuring your AppSec program, starting with how to use metrics to understand what is working and where you need to improve, to identify and solve problems, and to build a case for making further investments in your program. Ultimately, the goal is to make AppSec part of the organization's culture, and ensure it's relevant to business units and meaningful to executives.

Types of Metrics

There are three types of measures you need to work with to assess the effectiveness of your AppSec program. They are represented in Figure 1 and include:

1. **Technical:** Base data that matters to you and your team for detecting problems and defining processes, but is not interesting to people outside of your programs.
2. **Operational:** Data that helps you establish baselines and goals, identify problems, and determine where to focus and where you need to improve. These metrics are critical to properly lead and direct an AppSec program; they matter to your team, your management and the people you are trying to help.
3. **Executive:** Data that links business goals, priorities and business programs within the larger narratives of the organization. This data matters to executives and business units that can play a central role in the success of your AppSec program.

Figure 1. Measurements in an AppSec Program

Technical Metrics

Begin by collecting data on code vulnerabilities. This data should include:

1. **Number found.** A fundamental metric is the number of vulnerabilities in your code over time. This includes bugs found through threat modeling and code reviews; by static analysis security testing (SAST), dynamic analysis security testing (DAST) and interactive application security testing (IAST) tools; and through pen testing and other testing.

2. **Density.** The number of vulnerabilities divided by lines of code or some other proxy will give you vulnerability density, which makes it easier to compare risk in different systems, by technology platform or language and over time. Just like any other kind of bugs, some systems will have more vulnerabilities than others, and they will tend to cluster in certain areas of code.

Track production security incidents and breaches separately from problems found in your testing or reviews. Incidents and breaches indicate a failure of engineering and your security program. A good AppSec program should result in fewer incidents and breaches against applications.

3. **Severity.** Evaluate how serious the vulnerabilities are by determining risk by likelihood (discoverability, exploitability, reproducibility) and impact. Standardize risk scores across tools and applications, using a scheme such as the Common Vulnerability Scoring System (CVSS[1]), which ranks vulnerabilities from critical to low based on how easily a vulnerability can be exploited and its potential impact on data confidentiality, integrity and system availability. Weed out false positives, low probability findings and informational noise. Focus on real-world risks that could lead to runtime errors or crashes, breaches of sensitive data, fraud or compromise of critical systems.

4. **Type.** Use a common vulnerability taxonomy, such as the OWASP Top 10 web application vulnerabilities[2] or the CWE/SANS Top 25 software errors,[3] to help point out what is broken in the software and in the software development lifecycle. This process is the start of root cause analysis, which determines where to look for the source of the problem and how to halt it.

5. **Discovery.** How did you find the vulnerabilities? This will help determine whether vulnerability management processes—such as threat modeling or a code review, functional testing, or a code audit, pen test, DAST scan, SAST scan, bug bounty or vulnerability feed—are working. What you don't want is to report that discovery (gulp) occurred through an incident/breach—or even worse, have it reported to you by a third party.

6. **Number fixed.** How many vulnerabilities have been fixed?

7. **Time to repair.** How long did it take to fix the vulnerabilities? Or, to look at this data another way, how long, on average, did vulnerabilities stay open, especially serious vulnerabilities? What is your window of exposure?

[1] www.first.org/cvss

[2] www.owasp.org/index.php/Top_10

[3] www.sans.org/top25-software-errors/

This live data needs to be acted on, shared and used. Don't keep it in PDFs or a spreadsheet. Make sure to get it back to the people who need to work with it, and be sure it's in the format they need. Get it out of your tools and reports and get it to your developers' development backlog.

Don't just track application vulnerability data. Act on it.

Make this feedback loop as tight and automated as possible. Take advantage of APIs provided by automated security testing solutions and vulnerability management systems that tie into developer application lifecycle management solutions or directly into integrated development environments. The faster you can find real problems and get them back to developers, the faster—and more likely—these problems will get fixed. Speed is especially important if you are working with fast-moving Agile and DevOps teams.

Good Resource

The OWASP Testing Guide Introduction is a good resource for testing methodologies, tools and metrics.[4] It and other resources will prove that fixing vulnerabilities early in development is the least expensive and least disruptive course of action. Plus, there is no risk exposure if applications are scanned and patched prior to production.

Operational Metrics

Detailed vulnerability data by itself is useful, but it's not enough. You need to create operational metrics for you, your team, your boss, and the development/engineering teams and operations teams that you work with, as well as their managers and sponsors. Operational metrics enable everyone in this chain to see patterns, identify weaknesses and strengths in your application security program, understand risks and trends, and help make decisions and improvements.

You build operational metrics by slicing, dicing and rolling up technical metrics—in our case, application vulnerability data—to create different views, such as:

- Application/system type and criticality of the system (application portfolio view)
- Line of business or product line or region (organizational view)
- Whether the software is developed in-house, COTS (commercial off the shelf) or outsourced (ownership view)
- Programming language or platform (technology view)
- Regulatory and governance constraints (compliance view)
- Whatever other attributes or views you find interesting or valuable

Analytics dashboards from enterprise scanning tools and service providers are good places to derive a lot of this information. It may require middleware or additional vendor tools and services to bring it all together.

After you've collected this data, use it to assess the risk of different systems, and the health and effectiveness of your AppSec program. This data collection should be a continuous process, per the CIS Critical Security Controls and other frameworks and benchmarks.

Use this information to ask questions—then go find the answers.

[4] www.owasp.org/images/5/52/OWASP_Testing_Guide_v4.pdf

[5] www.cisecurity.org/critical-controls.cfm

[6] www.dhs.gov/cdm

Ask Questions

Asking the right questions will uncover dark spots where you don't have good data, or any data at all, and ferret out the risk areas and potential compliance violations not being addressed. Then, correlate the answers against other metrics data to determine improvements in visibility, risk, vulnerability repairs, organizational improvements and departmental comparisons.

Figure 2 presents a few examples, and a list of many more follows.

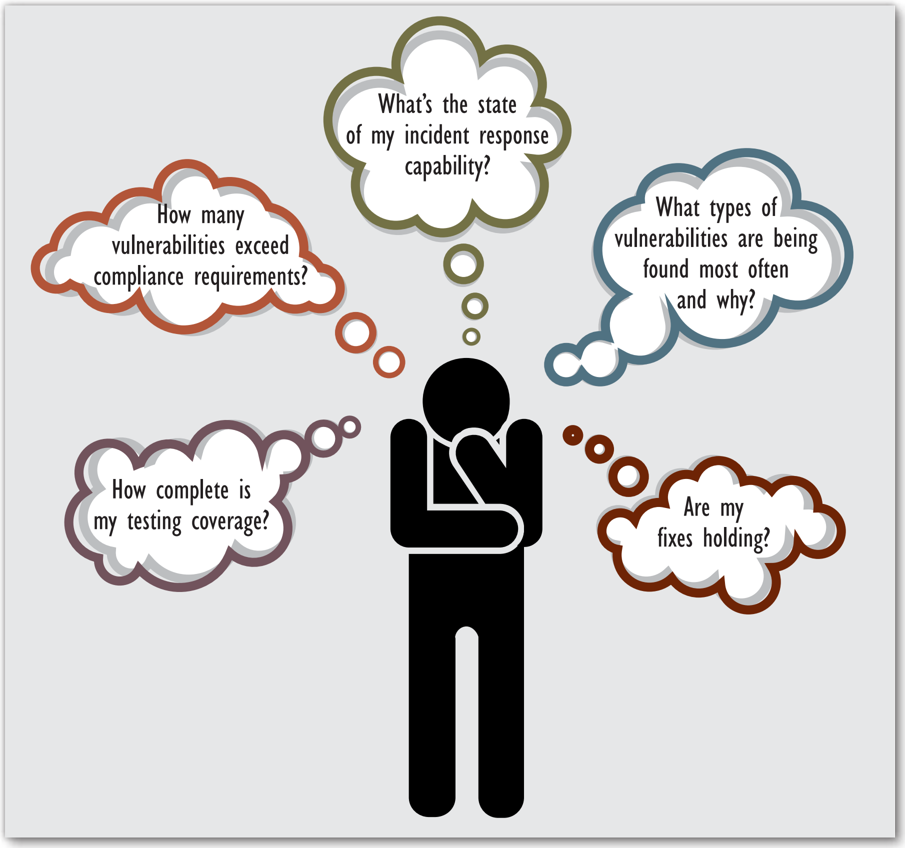

What's the state of my incident response capability?
How many vulnerabilities exceed compliance requirements?
What types of vulnerabilities are being found most often and why?
How complete is my testing coverage?
Are my fixes holding?

Figure 2. Examples of Questions to Ask About Your AppSec Program

1. Visibility

- What can you see? What can't you see?
- Are you testing all the right systems? Do you have a complete picture of all the applications being used in your organization, what they are used for, and what data they hold? How can you be sure?
- Which systems are being regularly assessed and scanned, and which aren't? What should you be scanning that you aren't?
- How complete is your testing coverage?

2. Risk

- Which systems have the highest number of vulnerabilities—or the most serious vulnerabilities and why?
- How many serious vulnerabilities do you have open and not repaired or patched yet? How long have they been open—that is, what is your window of exposure to attack? This data can be used to set a maximum window for serious vulnerabilities: 30 days or 90 days depending on your organization's tolerance for risk and compliance obligations. How many vulnerabilities exceed that window?
- What are the most common types of vulnerabilities being found? Why?
- Where are these vulnerabilities? It's important to track whether vulnerabilities are occurring with in-house code, third-party code or open source components.
- How are vulnerabilities being found? And when? This information can help determine whether some tools or practices are more effective or more efficient than others at discovering vulnerabilities early or later in the application life cycle.

3. Repairs

- What kinds of vulnerabilities are being fixed? What kinds are not getting fixed? Why?
- Are some bugs easier or harder to fix than others? How much does it cost to fix them?
- Are some teams able to fix problems faster than others? Are some teams not keeping up? If not, why not?
- Are the fixes holding? How many regressions are happening?
- Have you mapped the value stream from when a vulnerability is discovered to when a patch is fully deployed to production? What is the cycle time and the change lead time? What are the steps? Who is involved? How long does each step take? Where do delays and inefficiencies come from? What can you do to help improve this process?
- Are there other ways to close off vulnerabilities that may be faster or cheaper—especially for legacy systems or COTS—such as using WAFS with virtual patching or Runtime Application Self-Protection (RASP)?

4. Improvement

Look at trends on vulnerabilities found, when and how they were found, how quickly fixes are being made and deployed, and how long vulnerabilities are being left open.

- Are you seeing certain kinds of vulnerabilities increase or decrease over time? Are you seeing new risks or new types of vulnerabilities?
- What do you need to do to improve these trends? Do developers need more training? Should you be investing more up front in design, creating secure-by-default frameworks and secure design patterns? Do you need better, faster tools for earlier testing and feedback? Do you need more buy-in from engineering managers and business owners to make security a higher priority?
- What mistakes are you making? Are you learning from these mistakes?

5. Comparison

- Compare metrics between lines of business, development teams and types of systems. Create heat maps that show where most violations are occurring and scorecards on how effectively they're being remediated. Use this to create competition within the organization.
- Compare against industry data sets. A number of application security service providers regularly publish industrywide metrics and analysis that you can use to benchmark your program and risk profile. Where do you stand compared with your competitors?

6. Response preparedness

- If your organization comes under attack, or if you need to respond to a data breach, how quickly can you figure out what's wrong and fix it?
- What's the state of your incident response capability?
- Does everyone involved—including developers and their managers—understand what to do in the event of a security incident? Have you tested this understanding?

7. Compliance

Compliance is an important driver for AppSec in many organizations—often the most important driver—and therefore needs to be measured. Evaluate your program against regulatory requirements. Then determine whether you are doing enough.

- How many audit findings have you had, and in which systems? What are the consequences of being in violation?
- What gaps do you have in your controls or assessments?
- How many vulnerabilities exceed compliance requirements?
- What changes are coming up in regulatory requirements, and are you prepared to deal with these new requirements?

Keep asking questions until you get answers you can act on. Use your metrics information to build up a compliance scorecard. See Figure 3.

Figure 3. Scorecard of Compliance Issues to Address in an AppSec Program

Executive Metrics—Taking AppSec to the C-Level

You've got data from the first step. You've used this data to build insight into operational issues. Now it's time to introduce application security awareness and metrics to senior management and the board of directors. Unless you are working through a major breach response or investigation, most executives won't want to know about specific vulnerabilities and the details of your AppSec initiatives. And they don't need to know.

What C-levels (CxOs) care about is risk exposure to the success of critical operations, programs and customer initiatives. They care about regulatory compliance issues and legal risks. And they want to know how you can help solve important organizational problems.

When it comes to application security, CxOs want to understand:

- How has the application security program improved security risk posture overall?
- How are you helping to improve the availability and operational risk of critical systems and services?
- How are you helping to reduce time to delivery for important programs or cost of operations for existing services?
- What serious risks remain to critical operations and key products or programs?
- How do these risks translate to threats to key customers and company reputation?
- What are you doing to systemically reduce these risks?

Consider Their Motives

Don't rely on FUD (fear, uncertainty and doubt) to raise awareness among your business units and executives. And don't dictate, "We have to do this because compliance rules say we have to do it." If you use either of these tactics, you'll miss your chance to engage with line of business managers and operations executives who can be powerful sponsors and allies. Use compliance as a hammer only when necessary.

To make AppSec relevant to CxOs and business units, you need to make security a part of larger initiatives, programs and issues that are important to the business. Tie your application security programs to the organization's priorities and realities. Understand what is most important to your audience: Does the organization need to move faster and innovate, or save costs and increase efficiency? What customers or programs are critical to success?

Table 1 outlines executive priorities to tie your application security programs to.

Table 1. AppSec Program Priorities for Each C-level Executive

CISO	Tie into the overall Information security program and operational risk management.
CIO	Critical operations, high-profile programs, cost of delivery/service and cost of downtime.
CFO	Costs and savings, and how you are helping to reduce other costs. Costs as a percentage of overall IT costs or security costs or development costs, as well as costs over time.
CCO	The chief compliance officer should be a natural ally and sponsor—if you are doing your job. Prove that you are doing what you need to do to achieve compliance, or explain what you need to do a good job.
CEO, COO, line executives	Risks—and opportunities—across product/business lines, key programs and customers. Roll your risks up into the enterprise risk management program.

Point to success stories to prove your ideas and capabilities; build a quick win to show that what you want or need is achievable. Keep your reports simple and to the point, and focused on the lines of business that you're working for. Don't overwhelm those you report to.

Present problems and offer solutions. Problems without solutions are just problems nobody wants to deal with. Get executives involved in choosing the solution, to secure their buy-in down the road. Give them a chance to play the hero in your story.

Always Stay Aligned with Strategy

Make sure you understand the role you need to play in organizational strategy, including when strategy and priorities—and risks—evolve (which is pretty much continuously).

Organizations (or parts of the organization) looking to outsource, insource or offshore more work present different risks to development and application security programs. Cloud, mobile, social media and the Internet of Things introduce new risks and opportunities. It's important to plan for the effects these initiatives have on your program, and understand the risks that need to be managed and how to minimize those risks and costs and improve scalability.

From the C-level to the administrative level, costs will always be an issue everybody understands and cares about. If minimizing costs is the priority, metrics need to prove whether your investments are effective and where you need to increase efficiency. This could mean reducing cost of delivery by automating and streamlining security assessments, providing earlier and better feedback to engineering, scaling out to the cloud or building more skills in-house. The objective is to continually control evolving risk while reducing direct and indirect AppSec costs in engineering and operations.

If increasing innovation and speed to market is a key priority, then you will need to adjust your program's initiatives to balance operational risk and compliance with delivery speed. Understand how you can tie in to Agile, Lean, and DevOps initiatives focused on reducing the time and cost to delivery, while minimizing security and operational risks.

Look Inward. Rinse and Repeat.

Metrics are fundamental to understanding risk, measuring and demonstrating compliance, and driving toward tactical and organizational results. Use this information to gain insights into strengths and weaknesses, and to identity the root causes of security problems and how to prevent them.

After your metrics program is working, you should start to look inward, measuring and evaluating your processes and practices. Leverage application security program maturity models such as OpenSAMM[7] or BSIMM[8] to understand, measure and compare your program initiatives against best practices and industry leaders.

Use the metrics data you collect and analyze to find out what questions are useful and important. Then get the answers and build insight to improve your application security program. Rinse and repeat.

[7] www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model
[8] www.bsimm.com

About the Author

Jim Bird, SANS analyst and lead author of the SANS Application Security Survey series, is an active contributor to the Open Web Application Security Project (OWASP) and a popular blogger on agile development, DevOps and software security at his blog, “Building Real Software.” He is the co-founder and CTO of a major U.S.-based institutional trading service, where he is responsible for managing the company’s technology organization and information security program. Jim is an experienced software development professional and IT manager, having worked with high-integrity and high-reliability systems at stock exchanges and banks in more than 30 countries. He holds PMP, PMI-ACP, CSM, SCPM and ITIL certifications.

Sponsor

SANS would like to thank this paper’s sponsor: