VERACODE

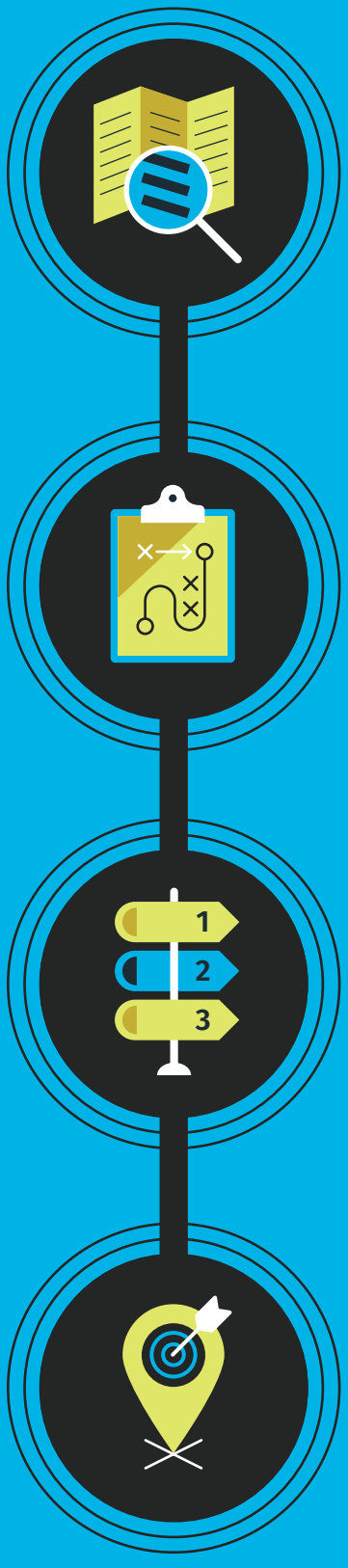
From Ad Hoc to Advanced Application Security

Your Path to a Mature AppSec Program

BY COLIN DOMONEY

Former Manager of a Global Investment Bank's Application Security Program

CONTENTS



Introduction

In a recent Verizon study:

OVER

375,000

Incidents

OVER

17,000

Data Breaches

ALMOST

40%

Resulted Directly from Web App Attack

by far the largest category

DESPITE THESE FINDINGS, IN A RECENT STUDY, ONLY HALF OF BUSINESS LEADERS SURVEYED FULLY UNDERSTOOD THE RISK THAT VULNERABLE SOFTWARE AS A WHOLE POSES TO THEIR BUSINESS.

Clearly, application security is misunderstood. And a good portion of the misunderstanding stems from not knowing where to start, or even what good looks like. We've worked with numerous companies on their path from zero AppSec to a mature, comprehensive program. I myself, in fact, collaborated with Veracode to start and grow an application security program from the ground up at a global investment bank, and now I'm happy to share that experience with you.

To shed light on how to get started with application security, and on what good looks like, we'll outline the steps most of our customers take to develop a mature application security program, and I will additionally share what each step looked like in my own application security journey.

AppSec Stages

Today, most organizations that have begun down the application security path are at one of the following stages. However, regardless of which stage you are currently in, your goal should be to move toward a mature, comprehensive program, which is ultimately the most effective way to protect your application layer.

Reactive Approach

Organizations taking a reactive approach to application security are typically driven by the need to comply with industry-specific regulations or specific security attestation requests from customers, and their efforts are reactive in nature. With this ad-hoc approach, organizations conduct security assessments outside the development lifecycle, typically so late in the process that a release will occur no matter the outcome of the testing, otherwise the release would be significantly delayed.

In addition, they assess applications using some form of manual penetration testing, either from internal teams or, more likely, by hiring a vendor to conduct a manual penetration test. This makes it difficult to scale without significant budget increases. Remediation is based on the needs of the customer or industry regulations, and only fixes the most egregious software flaws. With the move to Agile, DevOps and CICD, this is no longer a practical method to use.

PROGRAM MUST HAVES	SCORING	COMMENTS
Scale	✕	Focus only on top critical apps
Speed	✕	Days to weeks
Risk Reduction	○○○	Some on top critical apps
Automation	✕	None
Integration into SDLC	✕	None
Cultural Acceptance	○○○	Effect is minimal overall
Developer Involvement & Education	✕	None
Multiple Testing Techniques	✕	None
Central Governance & Reporting	✕	None
Cost Effective	✕	Very expensive with no scale

✕ NEGATIVE
○○○ NEUTRAL
✓ POSITIVE

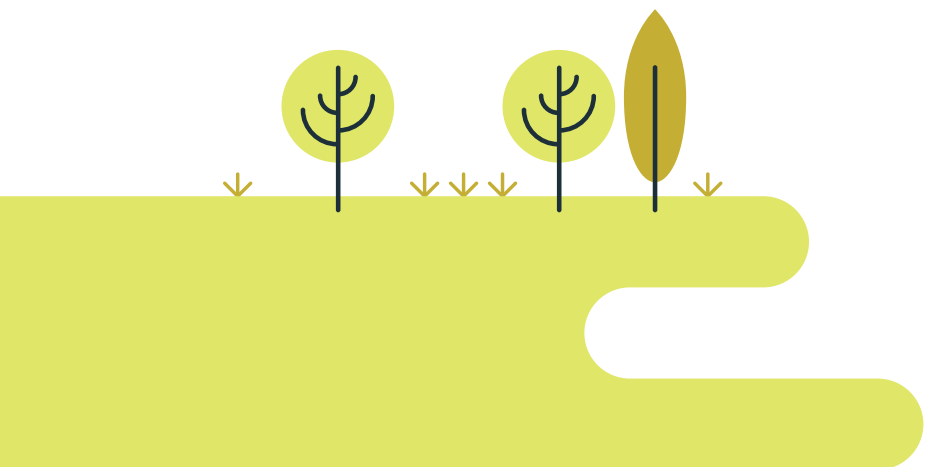
Baseline Approach

A baseline program covers a larger portion of an organization's application portfolio, but with a focus on business-critical applications. With this approach, application security assessments combine techniques such as manual penetration testing and dynamic analysis (DAST).

However, a security team conducts these assessments only at the very end of the software development lifecycle (SDLC), right before going to production. Finding these flaws so late in the development process is very costly to fix (10X more costly then catching them early in the SDLC). With the move to Agile, DevOps and CICD, this approach as well has become unpractical.

PROGRAM MUST HAVES	SCORING	COMMENTS
Scale	✕	Focus only on most critical apps but many can bypass this process due to deadlines
Speed	✕	Days
Risk Reduction	○○○	Some fixes late to post production
Automation	○○○	Some automated tools used
Integration into SDLC	✕	None
Cultural Acceptance	✕	Developers see this as slowing down their process and don't feel responsible for owning security
Developer Involvement & Education	✕	Developers are handed flaws to fix with minimal guidance
Multiple Testing Techniques	○○○	Dynamic and pen testing
Central Governance & Reporting	○○○	Some centralized reporting for security
Cost Effective	✕	The later flaws are found, the more expensive

✕ NEGATIVE ○○○ NEUTRAL ✓ POSITIVE

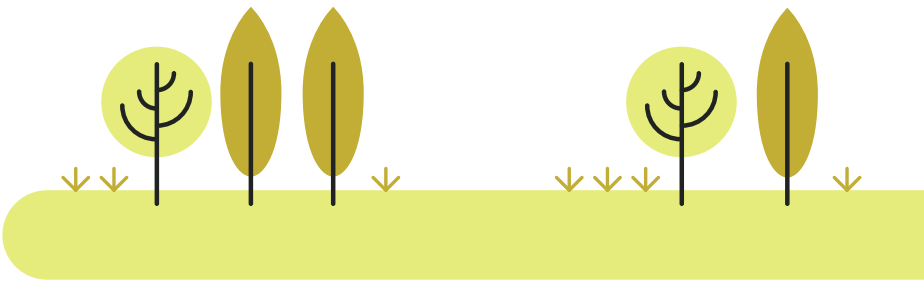


Expanded Approach

With this approach, security assessments are embedded into the SDLC at various stages of development. Many companies take this approach today. For security, this gives them the ability to make sure applications are securely built throughout the development lifecycle and provide developers with exactly what they need to fix, and keep building. Some organizations even have experts in the security team to help the developers. This is a solid and effective approach to application security, and will dramatically reduce risk. However, with the number of applications exploding and the pace of development rapidly intensifying, this approach will struggle to keep up and to scale. Developers are starting to consider this approach "scan and scold" and finding it slows their processes. In addition, security never has enough budget or people to scale this operation.

PROGRAM MUST HAVES	SCORING	COMMENTS
Scale	✕	Security can't build a team large enough
Speed	✕	Multiple touch points in SDLC that aren't automated
Risk Reduction	○○○	Some fixes on apps that are tested
Automation	○○○	Some automated tools used
Integration into SDLC	○○○	Some dev teams will allow for tools into their process to appease security, most won't
Cultural Acceptance	✕	Developers see this as "scan and scold." Security is intruding on processes they own, and then telling them their code is "ugly."
Developer Involvement & Education	✕	Developers are handed flaws to fix with minimal guidance
Multiple Testing Techniques	✓	Static, dynamic and manual based on app and stage
Central Governance & Reporting	○○○	Some centralized reporting for security
Cost Effective	✕	Security team can't hire enough people to make this work

✕ NEGATIVE ○○○ NEUTRAL ✓ POSITIVE



Advanced Approach

The advanced approach is the most comprehensive of the four, and scales to protect each and every application, regardless of origin (internal or external), type (web, mobile or legacy) and whether the application is considered business critical. This approach also considers the security of applications in production.

These programs have centralized governance by security, but testing and fixing are owned by development in an automated fashion throughout the build process using static and dynamic testing. In this approach, security owns setting policies, tracking KPIs and providing security coaching to developers. In addition, security owns providing developers with support in integrating scalable tools like Veracode into their SDLC. Developers own testing applications in their development environment, fixing flaws to pass policy and continuing to build code — no matter the environment: Agile, DevOps, CI/CD.

PROGRAM MUST HAVES	SCORING	COMMENTS
Scale	✓	SAAS instant on and unlimited users a must
Speed	✓	Minutes, as testing occurs seamlessly throughout SDLC
Risk Reduction	○○○	Policy is set by security so developers have a goal to reach for every test
Automation	✓	Solution must make testing unobtrusive
Integration into SDLC	✓	Developers own testing in their build process
Cultural Acceptance	✓	Security becomes everyone's job and a team effort
Developer Involvement & Education	✓	Developers discover flaws to fix and have guidance from the security team as needed.
Multiple Testing Techniques	✓	Static, dynamic and manual based on app and stage
Central Governance & Reporting	✓	One tool set that gathers all KPIs including policy compliance
Cost Effective	○○○	Developers now own testing and remediation, so a smaller security team can focus on value-add items like governance and coaching

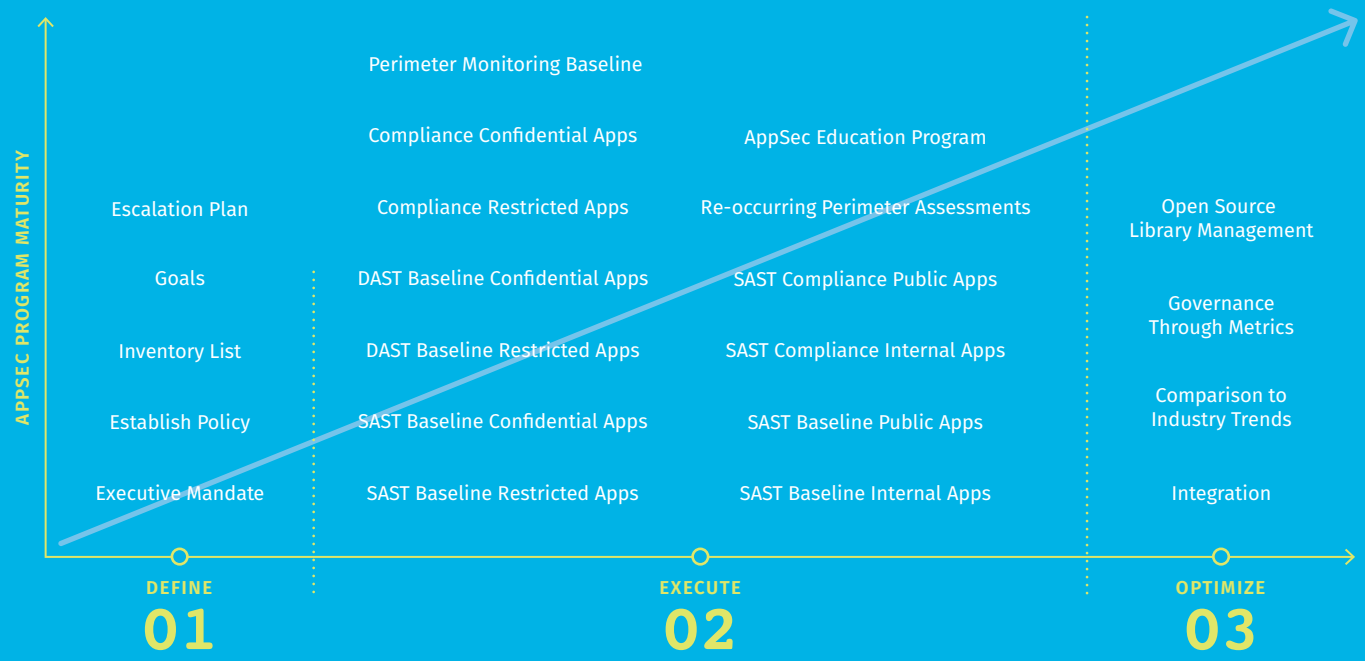
✕ NEGATIVE ○○○ NEUTRAL ✓ POSITIVE

Steps to Reach AppSec Maturity

A mature application security program might seem intimidating to some organizations. But it's important to remember that there are an established series of steps most organizations take when developing an application security program.

The keys are to start small, keep things simple, prove the value and then mature the program over time. In fact, the most successful companies we've worked with have started by securing a few apps at a time. In addition, if you build security assessments into the development process, reaching maturity is less daunting.

Below are the steps most of our customers take on their path to application security maturity, along with highlights of what each step looked like in practice, based on my experience managing application security for a global investment bank.



01. Define/Discover Step

In this step, you will define your program and communicate the mission internally. The goals of this step include:

Gain commitment from executive level — and communicate the executive mandate on AppSec.

The executive team's main concern around any new initiative is how it will impact the bottom line. It is the executive team's responsibility to ensure the company is operating efficiently as well as securely. If you have support for your application security program from the executive team, other departments in the organization will be compelled to participate and support the program as well.

Complete maturity assessment.

Veracode uses a variant of the OpenSAMM software assurance maturity model to help our customers to understand and improve their application security processes and posture.

Tip

It's incredibly easy to set the bar too high. While a high standard of security is vital, unrealistic expectations and requirements too often lead to people looking for ways to get around policies so they can get their work done. A basic fact of application security is that any policy should be only as complicated as it needs to be to deliver the necessary results.

A more realistic approach starts with attainable goals — such as the eradication of a specific high-risk threat — and expands from there.

Create an inventory of all your applications.

By running a discovery scan of your web perimeter, you can quickly gain an inventory of the most critical and easily exploitable vulnerabilities. From there, you can immediately reduce risk by either patching vulnerable sites or even eliminating sites that are no longer in use, but still active.

Establish program goals.

The most common strategy for AppSec goal setting is to use the OWASP Top 10 as a guide for vulnerabilities that must be remediated. Another metric that an organization can use is to baseline the organization's average application flaw density and set a goal around reducing the flaw density by a set percentage each quarter. Whatever the metric, it is crucial to first baseline the current status of application security at the organization and set predictable timelines for measurement frequency, as well as set expectations for what constitutes success and what indicates a need for continued improvement.

Define policy.

Scanning code for vulnerabilities represents only part of the solution. You also need a strong AppSec policy to increase the odds that an enterprise adheres to regulatory compliance requirements, as well as industry standards, and that teams are working in a consistent and synchronized way.

Define/Discover Step In Practice

I previously managed application security at a global investment bank, building an AppSec program from the ground up. Here is a snapshot of what Step 1: Define/Discover looked like for my program.

Highlights

Within the first nine months, we scanned 150 applications containing more than

100,000 **HIGH-SEVERITY FLAWS**

Developer-driven, ad-hoc remediation at this step.

<40% **AVERAGE APPLICATION SCORE**
Based on cumulative flaw count.

A relative indicator of application security based on cumulative flaw count. The average industry score is between 60% and 75%.

80% **OF APPLICATIONS WERE NOT COMPLIANT**

With our fairly basic policy, namely no high or very high flaws (no XSS, SQLi, CMDi).

Lessons Learned

What Went Well

Getting stakeholder buy-in.

- Identified business owners of affected applications.
- Identified application leads.
- Socialized program on internal websites.
- Held kick-off sessions in key sites.

Conducting an asset inventory.

- We figured out how the organization kept track of its application assets (who owns the apps, what is their business unit/owner/criticality/etc.).
- Got access to source control systems.
- Understood business criticality classification.
- Aligned with risk functions to agree on scoping.

What Went Less Well

Securing a mandate.

We should have done this earlier. In the first three years we relied on the co-operation of development teams to participate; they could have easily refused, and we would have had no rights to enforce participation.

Allowing application teams to be de-scoped too easily.

Once applications were excluded from the program, it was hard to bring them back into scope.

02. Execute Step

In the execute step, you will begin assessing applications and start remediation efforts, then move on to advanced testing methods and metrics analysis. Later in this step, you will introduce more advanced security processes. The goals of this step include:

Engage developers.

Development and DevOps teams' biggest fear when they hear their organization will enact an application security assessment program is that their development efforts will be slowed down. This team can be the biggest barrier to the success of the program, because if they do not follow the protocol set forth by the program plan, the security team will be unable to demonstrate the value of the plan. It's important to both understand the development process and then work to integrate security into those existing processes.

Initiate remediation coaching and education (eLearning).

Do your developers know what to do with scanning results? Do they know how to avoid introducing the same vulnerabilities in the future? Probably not, considering even the top computer science programs do not require cybersecurity classes. Help your developers create more secure code faster with training and remediation coaching.

Tip

We recommend measuring at least four application security metrics:

1. Compliance with policy
2. Flaw prevalence
3. Fix rate
4. A custom metric that aligns with your particular business goals

Data from our most recent State of Software Security report, reveals that best practices like remediation coaching and eLearning can improve vulnerability fix rates

BY AS MUCH AS

6x

Implement multiple testing techniques (for example, software composition analysis, static, dynamic).

There is no application security silver bullet. It is going to take more than one automated technique and manual processes to secure your applications. Only by gathering the strengths of multiple testing techniques along the entire application lifetime will you drive down application risk in your organization.

Over the past 10 years, we have scanned 2 trillion-plus lines of code, and we consistently see that different testing types are better an uncovering different vulnerabilities, and that one testing type is not enough.

Establish plan for collecting and reporting metrics.

The ability to collect and report on metrics is important in terms of compliance with regulations, understanding the performance of your application security program, and ensuring that you're reducing risk in the business.

Execute Step In Practice

Here is a snapshot of what Step 2: Execute looked like for my program.

Highlights

The goal of this step of the program was to expand it to include the top business-critical applications based on risk profile as follows:

- Applications in scope for Monetary Authority of Singapore regulations
- Applications in scope for Sarbanes-Oxley (SOX) regulations
- Applications that were public-facing on the Internet or exposed to a financial exchange
- Applications the bank deemed critical enough to require manual penetration testing

---

We rapidly expanded the scope of the program, leveraging SaaS and automation.

500+
APPLICATIONS SCANNED

500,000+
HIGH AND VERY HIGH FLAWS DISCOVERED

---

110 **APPLICATIONS**

Were on-boarded in a single month.

Remediation was developer-driven and ad-hoc.

Lessons Learned

What Went Well

- The policy was set to a very achievable level: no XSS, SQLi or CMDi.
- The CISO team partnered with the Engineering AppSec team to jointly drive the adoption.

What Went Less Well

- We had poor governance over the scanning process.
- Again, allowed application teams to be de-scoped too easily.

02

03. Optimize Step

During the optimize step, you will advance your application security program by implementing fully automated scanning earlier in the SDLC, assessing more vectors and using metrics to measure program success. The goals of this step include:

Integrate security into the SDLC.

The most scalable and practical way to ensure all applications built by an organization are assessed for security is to create an assessment process that is integrated into the software development lifecycle. By doing so, security becomes a part of the development lifecycle, rather than an afterthought tacked on right before the application goes into production. This increases efficiency, as remediating a vulnerability during the normal quality assurance processes is easier and more cost effective than doing so after the application's development is complete. Making the process seamless starts with integrating the assessment solution into the same APIs that are used for development.

Manage the use of open source libraries.

The best strategy is to gain complete visibility into all of the open source libraries development teams are using, as well as the versions being used. Consider adding technology that can help you keep track of your open source library usage into your AppSec mix. Only then can security teams ensure they can quickly patch and/or update the library version when a new vulnerability is disclosed.

Measure against established metrics.

When you establish metrics then measure your program against them, you not only prove that your program is making a positive impact, but also identify where and how it's working — or not working.

Our recent research found that approximately

97% **OF JAVA APPLICATIONS HAVE A COMPONENT WITH AT LEAST ONE KNOWN VULNERABILITY**

Refine goals.

AppSec is not one-size-fits-all; there are a variety of actors that will influence your particular threat landscape — including developer experience, programming language used and testing methods employed. With feedback about where and what vulnerabilities are emerging in your environment, you can tweak your system to address them more effectively.

Report against KPIs.

Key performance indicators (KPIs) can help an enterprise understand compliance, flaw prevalence, fix rates and business- and goal-specific performance. They provide insight into various factors, such as how many applications meet internal security policies and overall flaw density, and how these factors map to real-world costs. Streamline the process of getting executive buy-in and support by mapping your program to KPIs and reporting on the results.

Optimize Step In Practice

Here is a snapshot of what Step 3: Optimize looked like for my program.

Phase One

Highlights

Focus shifted to addressing the mounting technical debt.

We remediated several

100,000 **FLAWS**

OUR POLICY COMPLIANCE INCREASED BY >50%

Lessons Learned

How to Identify Security Champions

- Coach developers who self-identify as security experts.
- Incentivize within teams to encourage security champions.

What Went Well

- We took full advantage of Veracode remediation coaching.
- Started with "low-hanging fruit" to demonstrate success and establish process.
- Split effort into three categories: easy, medium and hard.
- Held training sessions with development centers on adoption.
- Developed a standard procedure for proposing, documenting and processing mitigations. This procedure required that each mitigation proposal include the following documentation:

  Technique: Type of mitigation in effect

  Specifics: Specific compensating control in effect

  Risk: Risk that mitigation does not address

  Verification: How was mitigation effectiveness verified?

Optimize Step In Practice (continued)

Phase Two

Highlights

Focused on embedding SAST into standard toolchain.

Performed automatic scanning of all production releases in the CI/CD pipeline.

>1,000

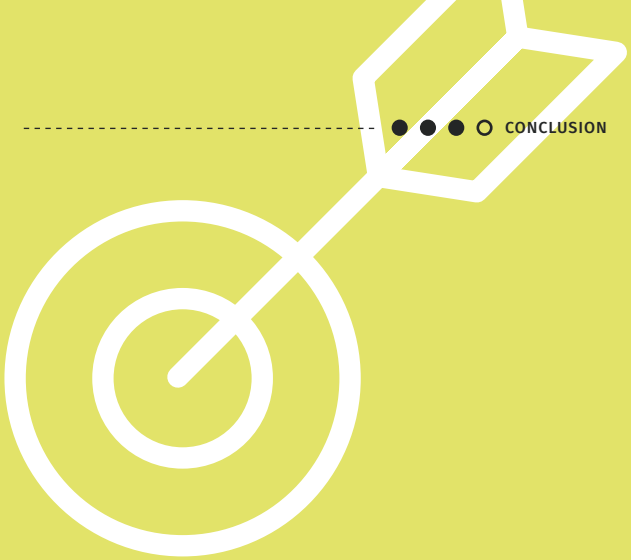
SAST SCANS RUN COMPLETELY AUTOMATICALLY EACH MONTH

The problem of insecure software open source libraries was addressed using Veracode's Software Composition Analysis product.

Lessons Learned

- Policy enhancements were made to ensure that application were scanned annually at a minimum and upon each major release.
- The scope of the program was widened to include all new applications entering the bank.
- We used Veracode APIs to automate scanning.
- We added an annual re-scan requirement to our policy.
- We decided that critical applications required a quarterly scan.

03



Conclusion

The application layer is far-reaching and fluid; you can't put up a wall or turn on a device to secure your apps. In the end, application security won't be sufficiently addressed with a one-off project.

But forward-thinking organizations are reducing their risk and moving their businesses forward with ongoing, comprehensive application security programs. Although creating this program might seem overwhelming at first, most organizations break it down into a series of manageable steps and slowly decrease their risk over time.

Learn More

If you want more details on the process, or help getting started or moving to the next step, please contact us.

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.

Learn more at www.veracode.com, on the Veracode blog and on Twitter.

VERACODE