A Practitioner's Guide to Application Security

The knowledge to build and improve your AppSec program using straightforward approaches that work

Caroline Wong
October 2018

Table of Contents

Introduction	3
Part 1: Getting Started	4
Application Security 101	4
Drivers: Why?	6
Talent: How?	7
Part:2: Diving In	9
Govern: What's Important?	10
Find: What's wrong?	13
Fix: What are we going to do about it?	15
Prevent: How do we scale our efforts?	18
Conclusion and Key Takeaways	22

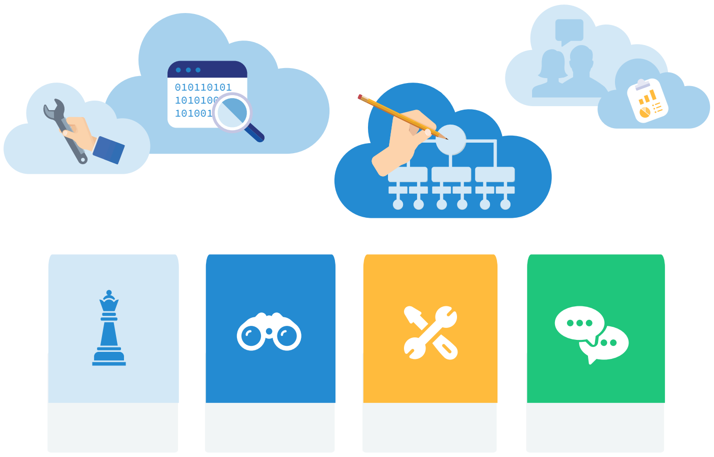
Introduction

So you've been tasked with building and improving an application security program. How do you go about it?

There are many frameworks and models that you could use (BSIMM, CSA CCM, ISO27017, etc.) but upon closer inspection, you might find these to be extremely long, overly complicated, and can be challenging to implement.

Web apps have become more complex, cloud apps are increasingly API driven, and code is being deployed faster and faster. The attack surface has changed and traditional application security has evolved.

Practitioners need a straightforward runbook to guide their application security efforts. That's why we created The Modern AppSec Framework.



Part 1: Getting Started

Application Security 101

Fundamentally, application security is about designing, building, and maintaining secure software. Good software helps organizations and bad software hurts organizations.

There are four main categories of application security activities - Govern, Find, Fix, and Prevent.

1. Govern

To do application security well, you must govern the application security program.

There are a number of high level factors to consider when you're thinking about application security. These include compliance regulations, relationships with other organizations, and having a solid understanding of what it is you're supposed to be securing in the first place. It's also important to define metrics up front so that you can demonstrate the success of your program over time.

2. Find

To do application security well, you must find security issues.

There are many ways to find security problems at different points in any software development lifecycle, whether your organization(s) follows a waterfall, agile, or DevOps methodology. Security problems exist in two broad categories - bugs and flaws. You can think of bugs as code level security issues and flaws as design level security issues.

3. Fix

To do application security well, you must fix security issues.

It is not good enough to just focus on finding security issues - the quality of software does not improve until the problems are addressed or eliminated. Fixing security issues requires effective communication, coordination, and integration with development teams and processes.

4. Prevent

To do application security well, you must prevent security issues from happening in the first place.

The people who build software must understand why vulnerable code is insecure. Developers must be empowered with tech stack specific knowledge and tools to help them avoid creating security bugs and flaws in the first place. Ideally, good programming practices and well-designed frameworks make it easier for developers to write secure software by default and harder for them to make mistakes.

Cloud environments must be configured correctly in order to prevent security vulnerabilities from being exploited, and attacks must be discovered and stopped as early as possible in order to minimize damage.

Drivers: Why do you want to do application security?

1. Sales / Competitive Differentiator

Maybe you've got a customer who insists that you perform application security activities because by working with you, they extend their attack surface and change their risk profile. Your application security is their application security. They want to maintain a certain risk posture, and you want their business.

You may be required to discuss application security practices as part of the sales process and provide evidence (such as penetration test results) before engaging in a formal business relationship.

Maybe the type of work that you do requires you to secure your software as a competitive differentiator. Security is a feature and promotes necessary trust with your customers and partners. Maybe your peers and competition are focused on application security, and if you are not then you might risk falling behind.

2. Compliance

Maybe you work in an industry that requires you to comply with regulations, and some of those regulations prescribe application security activities.

Examples of compliance drivers include PCI, HIPAA, GDPR, FedRamp, and SOC2.

3. Avoid a Breach

Maybe you (and your executives) don't want to see your organization's name in the news headlines because of a security breach. Maybe you don't want to have to snail mail all of your customers and tell them that their data was compromised by malicious attackers.

A security breach might damage the trust relationship you have with your customers and devastate your business.

Talent: How are you going to get it done?

If you don't have anyone on your team that knows application security, how exactly are you going to get the work done?

A holistic approach that matches application security talent and resources with the right technology is required to run the program. This is both a science and an art. Each organization will have business-specific context for different resource allocation trade-offs, program level metrics, and KPIs.

It would be convenient if you could “just” buy a piece of technology that would do everything for you. Unfortunately, application security scanners and firewalls can’t solve everything - their effectiveness is directly related to the skills and manual effort of people who must tune them to a specific environment, monitor them regularly, and filter the signal from the noise.

Organizations need human-powered security testing. But, it’s difficult to hire full time application security professionals. Consultants can be very expensive and may be unavailable for weeks or months, which doesn’t work if you need the job done right now. Useful security talent needs to snap into your agile development processes and produce results that align with your metrics and KPIs.

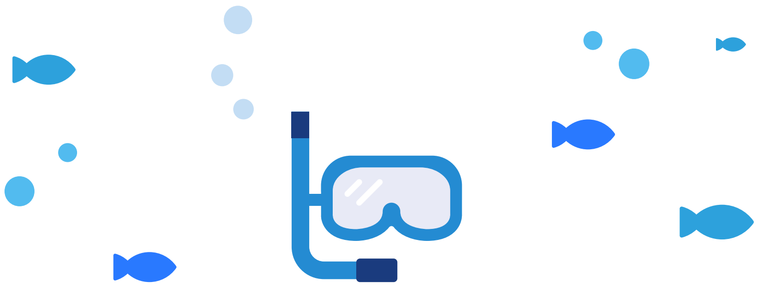
Today’s requirements for application security activities include:

1. Cost that enables higher frequency testing and greater coverage across an application portfolio

2. Access to quality security talent who can perform manual testing on-demand

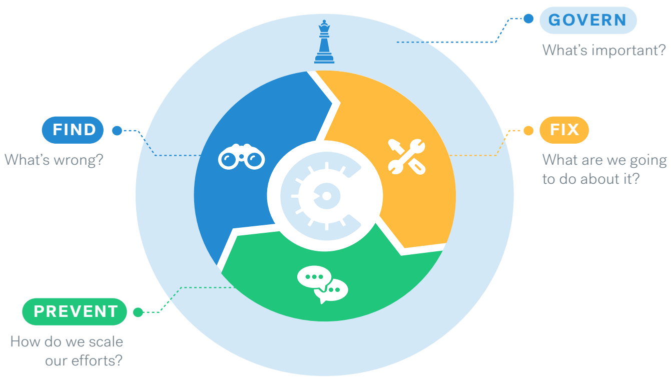
3. Strong integration with development processes and tools in order to get security issues fixed and prevented in the future

In short, today’s application security activities must be agile, actionable, and cost-effective.



Part 2: Diving In

This next section describes several different categories of application security activities that fall into each of the areas of Governance, Find, Fix, and Prevent.


GOVERN
What's important?
FIND
What's wrong?
FIX
What are we going to do about it?
PREVENT
How do we scale our efforts?

GOVERN	FIND	FIX	PREVENT
Asset Management & Risk Ranking	Pen Testing	Engage	Findings-Based Training
Compliance	Automated Scans	Communicate	Threat Modeling
Vendor Security	Secure Code Review	Integrate	Security Frameworks & Configuration Standards
Metrics	Vulnerability Disclosure	Track & Report	RASP & WAF

Govern
What's important?

Asset Management & Risk Ranking

If you want to secure your applications, the first thing to do is to ensure you have a reasonably accurate software inventory.

It's difficult to properly secure a piece of software if you don't know it exists or if you're not keeping track of it. Make a list of all of your web applications, mobile applications, APIs, and cloud applications. You also need to have an up to date, comprehensive list of various software components, their dependencies, software versions, and open source.

Next, assign a risk ranking to each piece of software. You can use criteria such as business criticality, data type, and accessibility to group your applications.

Important and high risk applications should have more (and perhaps different) security activities applied to them than less important and low risk applications.

Compliance

Many organizations are subject to security requirements because of an application's business function (ex. payments), type of data stored or processed (ex. healthcare data), or geographical location (ex. regional requirements for data privacy and protection). Others may be required to perform specific security activities due to contractual obligations.

Due to compliance requirements, certain security controls may not be optional. Find out what these are and make sure you're doing them to meet the appropriate standards.

Vendor Security

I like to say that "vendor security goes both ways." What I mean by that is that an organization's application security is affected by both the requirements of their buyers and by the risk profiles of their vendors.

An application that uses third party software components, including open source components, takes on the risk of potential vulnerabilities in those dependencies. These should be identified, tracked, and accounted for in the same way as every other software component (as described above in Asset Management & Risk Ranking).

If a software vendor is selling their application to a buyer, that buyer may require specific security activities (such as the results of a manual penetration test, a response to a vendor security questionnaire, or evidence of certain security policies) for the application.

Metrics

Best practices in application security change and go in and out of date very quickly. In application security, one size doesn't fit all. Standards and controls are built on years of practical security experience in real organizations – but this is something which is constantly changing. Today, every application security practitioner needs to know how to optimize his or her unique program using metrics.

Like any business initiative, an application security program should have objectives and measurements to determine if those objectives are being met. An example of a risk management objective for application security is, "Reduce the probability that attackers can cause critical applications to stop functioning." An example of a typical measurement that organizations track to help measure application security is defect density.

A security metric measures activity to provide decision support for doing things better in the future. This data can help to answer questions that an executive or operator might have about a particular area, such as penetration testing, using evidence-based information instead of opinion or anecdotes.

It's been said that "if you can't measure it, you can't manage it." While it turns out that Peter Drucker never actually said that, it is indisputable that measuring results and performance is crucial to an organization's effectiveness, and this definitely applies to application security.

Find
What's wrong?

Pen Testing

Penetration tests (aka pen tests) are a type of manual security testing that provides insight into an app's security by systematically reviewing its features and components. This type of exercise improves coverage of an app's security because the test is intended to explore the complete app rather than just focus on one type of vuln or one particular section. Pen tests follow methodologies related to topics like input validation, authentication, and access controls in order to identify flaws in the app's implementation.

Black box testers operate with limited knowledge and white box testers utilize as much information as they can to inform their approach.

Pen Testing as a Service (PTaaS) provides on-demand manual penetration testing for web applications, mobile applications, and APIs. Findings are delivered through a platform that integrates with developer tracking systems like JIRA and GitHub. A SaaS platform also facilitates collaboration between pen testers, security team members, and development teams to not only find but also to fix issues.

Automated Scans

Security scanners can be programmed to automatically identify certain kinds of vulnerabilities.

Application security scanners come in two flavors: A SAST scanner (“S” for “static” application security testing) examines the source code, binary, or byte code of an application. A DAST scanner (“D” for “dynamic” application security testing) examines the application from the outside when it is running.

The most interesting and important security findings cannot be discovered via automated means alone. Human intelligence and creativity is necessary to discover security flaws in business logic. There are entire classes of security issues (authentication, session management) which cannot be discovered using automated tools.

Secure Code Review

Code review is the manual review of one developer’s code by another developer. It’s intended to find mistakes and improve code quality. Similarly, secure code review is the manual review of code by a security expert. This is intended to find coding errors that may introduce security vulnerabilities.

Secure code review is a manual process that often leverages SAST technology.

Vulnerability Disclosure

Every so often, a security researcher that is not directly associated with an organization will discover and report a security vulnerability. This is called vulnerability disclosure.

Bug bounty is a type of vulnerability disclosure program which leverages a crowd of globally sourced researchers in competition. In a public bug bounty, anyone in the world can submit a potential security vulnerability to an organization, and the first to find a valid bug will be paid a “bounty.”

Fix

What are we going to about it?

Engage

The ways in which development and operations teams interact is changing, and security must keep pace. Security teams working effectively with DevOps teams, processes, and tools are absolutely critical to getting application security done right.

Security teams sometimes place heavy emphasis on finding issues, without enough focus on engaging with the development teams and building the cross-functional relationships that are actually required to get security issues fixed.

Fixing security issues is not just a technical problem; people and process are also required to get it done. Once you've performed security testing in order to find as many issues as possible, the next step is to engage with the teams that can actually fix the issues.

Get curious about development team priorities and look for areas of common interest. Ask questions about how development teams work and how much time they have to realistically spend on fixing security issues. Make sure you understand the business context for the security issues you want fixed, and use this information to prioritize fixes.

Communicate

Once you have performed security testing in order to find as many issues as possible, the next step – by no means a trivial one – is to communicate them to the development team. The development team is a critical stakeholder when it comes to prioritizing the fixes, remediating the issues, and ideally preventing the same issues from coming up again.

It's easy to say that we want to fix all the security issues that have been found, however it's not always easy to make it happen.

Developers are focused on building new features and meeting deadlines, and have limited bandwidth to remediate security issues. It is certainly not possible to fix all the security issues at once. They must to be prioritized in the context of business values and goals and addressed over time.

Integrate

One of the best ways to get security bugs fixed is to integrate with developer tools and processes.

Get curious about what tools developer teams use to do their work and the processes they follow to manage it. For example, how frequently do they release code? This should influence the frequency of security testing. What bug tracking system(s) are they using to manage bug fixes? Make sure security bugs are included and don't get lost in separate systems or PDF reports.

Customize the application security program to the way that your organization's development teams work and you'll see the most success in getting security issues addressed and eliminated.

Track and Report

Now that you've got your software inventory (Governance: Risk Ranking) and you've identified some security issues (Find: All), you'll want to keep track of what has been tested, by what means, and when. Keep track of the findings from each security test and prioritize them for fixing. Use business context to understand which issues matter the most, and work with development teams to fix those first.

For example, many organizations require that findings be fixed within a certain period of time, depending on the criticality of the findings. An e-commerce business might require that critical findings discovered on its customer facing applications be fixed within 48 hours, high severity findings be fixed within 10 days, medium severity findings be fixed within 30 days, and low severity findings be fixed within 90 days.

Make sure you always know which issues are open and which have been addressed and can be closed. Report summary information to relevant stakeholders so everyone is always in the know about current status.

Prevent

How do we scale our efforts?

Findings-Based Training

The best application security training for developers is based on real security findings, whether these are demonstrated during an actual security incident or found in manual penetration testing.

The OWASP Top 10 contains a list of common web application security risks, however each organization will have its own unique top 10 list. If you know what yours is, you can and should use this information to prevent entire categories of security vulnerabilities by implementing focused developer training.

Threat Modeling

There are two types of application security problems: bugs and flaws. Bugs are code-level mistakes, and flaws happen at the design level.

Threat modeling is a type of design-level security assessment that is intended to examine the way an application system works in order to identify potential flaws. The process involves analyzing assets, security controls, and threat agents in the context of an application system.

When flaws are detected in threat modeling before software implementation, some security problems can be avoided.

Security Frameworks and Configuration Standards

Some security issues can be prevented by using certain security framework and configuration standards.

A few examples include CSRF tokens (prevent Cross Site Request Forgery attacks), CSP (whitelist assets that the browser should allow to load and execute in order to minimize the impact of Cross Site Scripting exploits), and HSTS (encrypt data in transit and prevent fallback to non-HTTPS traffic).

Other kinds of security issues can be avoided by securely configuring the software environment, for example by following the Amazon CIS benchmark to harden AWS accounts and cloud services.

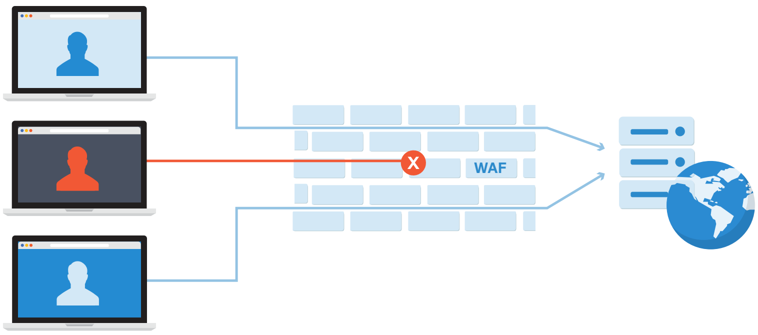
RASP & WAF

There are a couple of tools which are meant to protect an application by identifying and stopping malicious activity while the application is running.

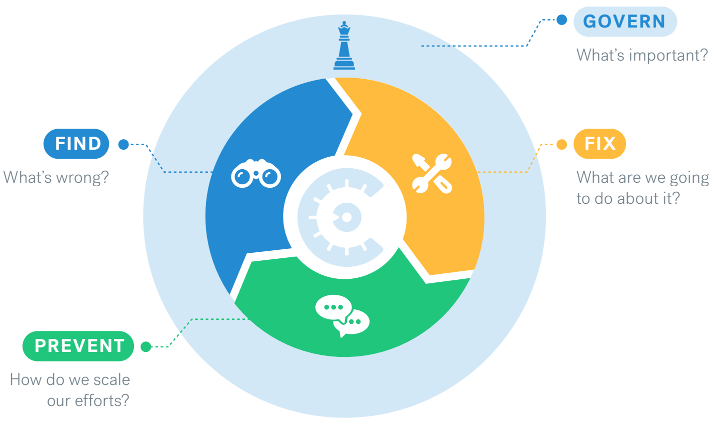
A Web Application Firewall (WAF) examines web traffic to identify and block suspicious activity, such as comment spam, XSS, and SQL injection attacks. Runtime Application Self-Protection (RASP) operates in the runtime environment and can monitor, detect, and alert in real-time.

Both WAF and RASP can be run in either “detect and alert” or “detect, alert, and block” mode. They can be most effective at preventing security issues when running in “detect, alert, and block” mode, however this requires the business to risk blocking legitimate application activity as well as malicious activity.

As with security scanners, any automated technology will simply do what it is programmed to do. The most effective implementations are carefully watched and guided by manual human effort as well.



The Modern AppSec Framework


GOVERN
What's important?
FIND
What's wrong?
FIX
What are we going to do about it?
PREVENT
How do we scale our efforts?

GOVERN	FIND	FIX	PREVENT
Asset Management & Risk Ranking	Pen Testing	Engage	Findings-Based Training
Compliance	Automated Scans	Communicate	Threat Modeling
Vendor Security	Secure Code Review	Integrate	Security Frameworks & Configuration Standards
Metrics	Vulnerability Disclosure	Track & Report	RASP & WAF

Cobalt
Pen Testing as a Service

Conclusion and Key Takeaways

It’s a great time to be in application security. The industry talent shortage means that options exist for professionals who possess and are developing skills in the field. It also means that finding the right people to help you build your application security program can be challenging.

The people who are going to attack your software aren’t just using technology to do it. They’re also using their smarts, analysis, and creativity. We need on-demand human effort to build and defend the software that powers our organizations.

Just like any other business initiative, an application security program takes a combination of people, process, and technology. Tools can be useful, but they need the right people and workflows to be effective.

Application security is very much a team effort. Security professionals can’t do it alone - they’ve got to work in collaboration with development teams.

We hope that you find this guide to be helpful as you develop and improve your application security program.

Caroline Wong
Chief Security Strategist
Cobalt.io

Her close and practical information security knowledge stems from broad experience as a Cigital consultant, a Symantec product manager, and day-to-day leadership roles at eBay and Zynga.

Caroline authored the popular textbook Security Metrics: A Beginner's Guide, published by McGraw-Hill in 2011. She has been featured in the 2017 and 2018 Women in IT Security issues of SC Magazine and was named one of the Top Women in Cloud by CloudNOW.

Special thanks to:
Julie Kuhrt for Art Direction
Mike Shema for Technical Review
Chris Tilton for Editorial Review