Secure Coding.
Practical steps to defend your web apps.

Copyright SANS Institute
Author Retains Full Rights

Interested in learning more?
Check out the list of upcoming events offering
"Defending Web Applications Security Essentials (DEV522)"
at http://software-security.sans.orghttp://software-security.sans.org/events/
Using Cloud Deployment to Jump-Start Application Security

A SANS Whitepaper

Written by Adam Shostack

May 2017

Sponsored by

Veracode
IT operations people have had it tough during the migration to the cloud the past several years. Most people in IT and operations have learned to manage their worries about the loss of visibility and control that happens during that shift. Organizations no longer think about a move “to” the cloud, but rather a move between cloud providers, between backend services, and how the new services world enables these shifts to happen faster and more transparently (or more opaquely—“What do you mean we’ve added three analytics vendors?”).

Security has learned to manage traditional apps as they move to the cloud, but doing it effectively requires more custom software, more integration software and more orchestration software. Those technical changes are accompanied by new skills and tooling to secure it all, and those mean new ways of working and new collaborators to work with.

As you move applications from your systems to someone else’s, things change. It’s possible to simply adapt to the cloud. There is far greater potential, however, in taking advantage of the move as an opportunity to re-engineer your approach to security and development in ways that can improve both and create greater organizational security.

Gunnar Peterson, author of one of the first practical blueprints for security in the cloud, has jokingly noted the remarkable consistency of IT security in relying on “firewalls, anti-virus and SSL” even as development advanced from HTML to CGI to Rest to Web 2.0. That approach no longer works.

So much new code is being written that on August 4, 2016, General Electric CEO Jeff Immelt announced that henceforth, everyone at his company would learn to code. Everyone. “It doesn’t matter whether you are in sales, finance or operations. You may not end up being a programmer, but you will know how to code.” While you may not work at GE, an announcement like this crystallizes a trend. Venture capitalist Marc Andreesen has said, “Software is eating the world.” That’s not only true in the world of atoms, but it’s also true in the world of bits. Amazon, Google, Facebook—all of the large operators of systems have moved from systems that are administered by people clicking and typing to systems that are managed by software.

The software that manages those management systems is also managed by software, but the chain doesn’t end there. That layer of software-management management-software is also managed—shocking as it must seem—by more software.

---

1 “Security Architecture Blueprint,” Arctec Group LLC, 2007, [https://pdfs.semanticscholar.org/2b20/5dc7ebbbf97a3c0928854fa09d8ba5d1e08d.pdf](https://pdfs.semanticscholar.org/2b20/5dc7ebbbf97a3c0928854fa09d8ba5d1e08d.pdf)

2 “Security Evolution,” [http://1raindrop.typepad.com/1_raindrop/2008/05/security-evolut.html](http://1raindrop.typepad.com/1_raindrop/2008/05/security-evolut.html)

3 “Why GE is giving up employee ratings, abandoning annual reviews and rethinking the role of HQ,” [https://www.linkedin.com/pulse/why-ge-giving-up-employee-ratings-abandoning-annual-reviews-immelt](https://www.linkedin.com/pulse/why-ge-giving-up-employee-ratings-abandoning-annual-reviews-immelt)
That automation is created and managed under a variety of labels, including DevOps and Site Reliability Engineering (SRE). The distinctions matter less than the commonalities, and those commonalities lead to organizational changes designed to enable rapid, secure development. They also drive rapid evolution in job descriptions that blur organizational roles to the point that it’s not unusual for operations to write code or for developers to carry pagers (Well, “pagers.”). Organizations face three interrelated challenges:

- Organizations must become comfortable building and maintaining software.
- Organizations must integrate security all through the development cycle, not simply “penetration test it in” at the end.
- Organizations must develop the maturity to understand the first two changes and implement them consistently.

If the first two changes were isolated to a few developers and a new tactical partnership with the security team, then executives would justifiably ask, “Why are you telling me about this?” But GE’s Immelt draws attention to the programming change because it’s a fundamental new reality. Executives are paying greater attention to security in the news and are asking, “What does it mean for us?”

All of this leads to greater understanding of the need for security at the executive level as well as at the technical levels. It also leads organizations to consider cloud not a threat to security, but an enabler of security. Now, let’s look at each of these changes.

**A New World Demands New Skills, Tools, Software and Leadership**

The integration of development, operations and security creates fundamental changes in practices and the skillsets security organizations need. The ability to analyze a packet trace, for example, must be complemented by the skill to write software that detects an abnormal netflow. The ability to take apart a core dump must be expanded with software to alert humans that core dumps are present, then copy them to a repository and open a tracking issue. All that software enables organizations to move faster and, as the collection of software that manages configuration, testing, monitoring, alerting and response grows, so does the organization’s ability to meet each new challenge faster.

And as that software becomes the mechanism that runs the business, the need to ensure appropriate security practices in the creation and maintenance of software becomes more critical.
If software is being created and updated at an ever-increasing rate, it makes sense to have software to help with that process. Actually, that’s a little bit of a lie. You need to have software to help manage the scale and speed of change. It makes sense to use the software that your developers use and to act as participants in their world. Today, those are tools such as git and Jira.

These changes require most people to learn new skills and competencies in programming and software development. It also implies that development will start to take up much of the inter-organizational effort that was previously devoted to operations. Of course, you can’t take your eyes off the relationship with operations as you partner with development.

This change will happen more smoothly if someone actively leads it. That active leadership involves using communication tools such as all-hands meetings and regular staff updates. You need to talk with your team about the journey and why you’re moving. You need to celebrate the accomplishments of the past as you look to different challenges for the future. You need to help people develop the hard new technical skills with training, projects to exercise those skills and rewards for “taking the plunge.”

These changes constitute a huge challenge for security and, in particular, for the person tasked with driving security for development.

More code is at the heart of this change. More software that runs the business will be developed locally. Some of it will be written by security staff. Some will also come from development as the center of gravity—the obligation to ensure that new code is secure—shifts away from operations and toward development.

The proliferation of cloud-based services means most organizations will rely less on traditional products and more on services and the orchestration among them. More of the software that secures the business will be developed locally. All of that code needs to be secured. We need to secure things across the development lifecycle, from concept to deployment to retirement/replacement, and we need to develop the maturity to ensure that we know what apps are in place and how to deal with the problems that inevitably arise.
Organizations fail to adapt to these changes in two major ways:

First, organizations fail to create an across-the-board mandate to ask what “software eating the world” means for each team affected and how well each is able to adapt.

Second, security gets caught up in some particular tool, skill or approach and fails to look strategically at what’s happening.

**Security Across the Organization**

As a security organization, you manage a portfolio of controls across your organization. One substantial part of that portfolio focuses on protecting the operational systems, and that part of the portfolio likely won’t go away soon. It’s also not likely to grow very much. What will take an increasing amount of attention is security for the application portfolio. Until here, the reasoning I’ve presented has focused on business drivers, but there are also two important security drivers. The first is the maturing of application security tooling. The second is the maturity of attack tools. The two toolsets interact, but they are not identical. A good attack tool tells you “here’s a vulnerability” and how to exploit it. A good defensive tool tells you “here’s the code you need to fix it” and maybe even “here’s a suggested patch.”

When you think about security for your currently operational systems, you need to do so in the context of the overall risk, which is likely growing, and the operational investment, which is likely shrinking. There’s a tension there. On one hand, security risk is growing. It’s growing because of new vulnerability discoveries in both your code and your dependencies, because new attack techniques make previously safe code unsafe and because applications that may have been “safely” behind a firewall are coming under scrutiny. That scrutiny is happening both because applications are being forklifted into the cloud and because of a growing recognition that your firewall doesn’t keep out all the bad guys.
Security in Application Development

If you’re using penetration testing broadly to drive security improvements, applying pen testing to operational systems like these can look arbitrary. It’s probably better to start early in the application development process by analyzing the apps or services on which your software depends and managing vulnerabilities in those dependencies. (Any software can have vulnerabilities. It may be easy or hard to exploit them. If you have good automated tests, it is faster and easier to update to a fixed version than to confidently analyze exploitability.) A simple way to start is to make a list of dependencies. NPM, RubyGems and similar tools explicitly list dependencies, and you can see who in your organization subscribes to the security-announce list for each dependency. If you’re doing that, a next step is to use predictable analysis techniques, such as static analysis or web application scanners. The advantage to using a technique like static analysis is that it’s predictable and manageable.

With respect to new applications, it’s easier to set modern security requirements as part of the delivery process. You can require the use of Safe Haskell by default, for example, for all new modules. Here, the advantage to using static analysis is that it can be easy to build into the development process from the start and, because it’s built in, it’s easier to plan for. In contrast to pen testing at the end of the process, static analysis doesn’t produce a single pile of bugs and issues that needs to be triaged and possibly addressed at the very end. And in contrast to either approach for apps that have shipped, there’s someone ready to look at the issues that program analysis can find.

It’s also helpful to choose modern languages and to select libraries that take security into account. Modern languages will often address issues that were visible in early languages: few languages have repeated C’s string handling issues, for example. Similarly, if your library makes it easy to write prepared statements, then SQL injection is far less likely to happen.

The best way to quickly look at a library to see whether it takes security into account is to look for the Core Infrastructure Initiative’s “Best Practices Badge.” That github badge is based on an evolving set of security criteria, such as having a security reporting process and supporting HTTPS. If you have a choice between a library with the badge and one without the badge, take the one with the badge. If you have a scorecard, checklist or other structured approach to choosing libraries, the badge makes it easy to add security. If you don’t, setting up a simple set of criteria is worthwhile. Those criteria might be as simple as “is the license acceptable; is there a security-announce mailing list; is the project alive and being maintained?”

---

4 Safe Haskell takes a more strict view of type safety and might be hard to add to existing code: https://downloads.haskell.org/~ghc/7.8.2/docs/html/users_guide/safe-haskell.html

5 You can find a list of software at https://bestpractices.coreinfrastructure.org/projects (Note that that link produces a list of all software that has started the process, and there’s a menu to select those that have passed or made certain progress.)
As you think about leading your team or organization in this changing world, it is useful to take a page from the National Institute of Standards and Technology Cybersecurity Framework (NIST CSF) and ask yourself questions about how you’ll inventory, protect and detect. You want to inventory and protect applications, and you want to detect security problems or issues. And you want to manage all of those. You should categorize all of your investments in application security this way even if you don’t use the CSF. Thinking of your investments in a portfolio helps you develop a strategic view. The NIST framework also addresses response and recovery, which are beyond the scope of this paper.

**Identify**

Identification is all about knowing what applications exist, where they are, what they consist of, who is responsible for their security and the like. Historically, security has often relied on scanners such as Burp or skipfish to help them find web applications. This reflects the penetration testing pedigree of the tools and approaches, and it aligns with the often distant or hostile relationships that have existed between security and development. Scanners can also serve as a useful check that applications are being deployed “in accordance with the rules.” You don’t want to abandon such tools, but it is time to augment or re-prioritize their use and to ensure security has access to the code repositories in which applications are developed and from which they are increasingly deployed.

Of course, your organization doesn’t only deploy its own code. Your organization’s code repository is a great place to start because you probably have the greatest ability to influence the code in it. But you probably deploy other applications: open source, closed source, cloud apps, etc. It is important that your inventory covers all of these code sources.

You can think of code created by others in two major buckets: code that runs on your infrastructure and code that runs on theirs (SaaS).

For code that’s not yours, but which runs on your infrastructure, you can work to improve it by looking for evidence of security development lifecycle work as part of your software selection process and you can work to limit its blast radius when it fails, using sandboxing, segmentation and other isolation techniques.

Security people often frown at SaaS. We can’t see into development or operations and, as a result, imagine the worst. But the people running a SaaS offering can often update faster and see more deeply into their product than can a similar on-premise package. Therefore, you should avoid the mistake of viewing SaaS as an automatic win or loss, but rather consider what practical tools are available to help you procure and operate it at the most appropriate level of security.
Protect

There are two times you need to invest in protecting your applications: before you deploy them and afterwards. Some of the tasks you engage in stretch throughout the application lifecycle but are different pre- and post-deployment. An example of something you do throughout the application’s life is dependency management. Static code analysis, however, is done before deployment, and log monitoring or analysis is done afterwards.

The most important steps, in approximate order of execution, are as follows:

- Design security, including threat modeling, good language choices and designing for defensible operations.
- Development security, including integrated testing, test-driven design, static analysis and fuzzing. (Fuzzing is often included in a testing phase, but increasingly, organizations are bringing testing and quality forward as they move to CI/CD practices.)
- Operational security, including secure deployment, configuration management and monitoring.

It is important to apply these practices consistently. That may mean applying them consistently across the portfolio. It could also mean responding consistently to specific levels of risk associated with each application—meaning what types of data an application handles or other risk factors for your organization. Thus, there might be “Priority 1” applications that handle patient medical data; “Priority 2,” which handle billing and credit; and “Priority 3” for marketing material. This apparently simple approach can get tricky pretty quickly—customer support might be Priority 3 until you wonder, “What happens if a customer asks a question about why there’s an HIV test in their record?”

It takes time and effort to develop the personal and organizational skills to be able to prioritize. Many security professionals know the marketing site acts as a stepping stone to the more sensitive data and want to ensure that everything is secure. That’s an admirable goal and means efforts can be spread thin. It also takes time to develop the skills to identify tricky areas and to manage them appropriately. Both are factors in maturity.
Detect

Let me start with an attractive distraction: Detection is not about producing log messages, alerts or alarms. It is about producing the right alarms, worthy of human attention. This is both easy to get wrong and hard to get right.

Getting it right is a matter of both technology and processes. You need a technological pipe to get all the messages through the right funnels to the right places. The process elements obviously include refining those pipelines, but, more importantly, they involve getting the right sorts of messages from your application portfolio: logins, security changes and behavior the developer doesn’t expect. Security changes can be as obvious as a password change or a change to email forwarding rules. This means you have to start thinking about detection early in the development process.
Organizations get good at the things they focus on. The maturity of a security organization is often treated as a poor cousin to more “important” technical security skills. Mature organizations—and the components that go into making an organization “mature”—are those able to deliver predictable, repeatable work, which leads to faster outcomes with higher quality. That’s why organizations make the investment. Remember: It’s easy to go overboard chasing maturity goals by focusing too much on the process rather than the outcomes.

Why Does Maturity Matter?

Consider two organizations: Acme and BigMegaBank. Acme’s artisanal engineers design and produce a variety of items, ranging from anvils to exploding wabbit traps. Acme’s ability to respond to requests from customers is legendary. Unfortunately, so are reports from customers indicating that there are problems with equipment meeting specs. BigMegaBank ships 20 to 40 new web applications per week. Each week, several fail acceptance tests and are pushed back by a week or more; however, the pace of shipping ensures new apps are always coming out and the delays are usually manageable.

At Acme, its highly regarded team of security specialists thinks carefully about the risks of each product from first principles and often identifies fascinating vulnerabilities in the design, which, after fixing, make for great content on the summer conference circuit. Because security talent is hard to hire and retain, Acme has never run a customer satisfaction survey to learn what its product engineers think of the security engineers.

At BigMegaBank, there is a structured approach to each project, starting with risk management, and a defined escalation path if tasks aren’t executed in predefined timeframes.

Which of these sounds more like your organization? Which one could your security process handle effectively?

As your security processes mature, a few things happen:

- **Your work becomes more predictable**, which makes it easier to interface with you. So, more teams work with you early, which means you process fewer exceptions and crises.
- **Your work becomes more consistent**, so the issues you address are more likely to be important ones.
- **What’s unusual becomes more visible** because there’s a stream of usual. This makes it simpler to escalate issues that are real risks.

As your processes become more mature, more consistent and predictable, management gets happier with your performance because the sorts of people who become management like things that run like well-oiled machines. Showing consistency and execution with the resources you have is a useful part of getting more resources.
Assessing Maturity

Classical maturity models such as the Capability Maturity Model Integration (CMMI) start at level zero or one “chaotic” and progress through stages, rating how repeatable and predictable a critical function is. The original CMMI has two levels of repeatability: one for the ability of a project team to repeat practices, the other rating how well an organization as a whole does it. There’s a trap that many organizations fall into, which is to get very focused on the minutiae of CMMI or some other maturity framework. One way to avoid that is to focus on why you’re doing the work, the goals you want to achieve and if you are achieving them, rather than on the choice of maturity standard or the precise definitions.

Sometimes even asking the question “How mature are we?” can lead you down a path of expensive consultants asking naïve questions and producing expensive slide decks. Or it can lead to people tussling over the way in which a word should be interpreted. While either might help the organization characterize where it is or assess if it’s in the right place, you might actually want to drive change.

So, here’s a “finger in the wind” test of maturity. State your answers out loud.

• Does everyone know how and when to get security involved in a project?
• Are the steps security will follow documented?
• Is there a clear way to resolve disagreements?
• Can you estimate how long it will take?
• Do you have software that helps you evaluate security consistently, application over application?
• Do you have a post-mortem process that helps you improve?

Give yourself a point each time you confidently answered “yes.” Give yourself “no” points if you hemmed, hawed or grunted. Do you want half points? Does that seem unfair? You’re right, it is. Please do feel free to go through a weeks- or months-long process of having some consultants from a big four strategy firm assess how you’re doing. And while that might give you a different result, the results you get from the six-question test are a good, quick view into where you are.
What’s the Right Maturity Level for the Organization?

Now that you have an idea where you are, you can ask what the organization needs. (Note: This is not “Where do you want to be?” but where the organization needs you to be. Maturity for its own sake isn’t productive.)

So, let’s circle back to that (somewhat arbitrary) test:

1. Does everyone know how and when to get security involved in a project? Addressing this will help you define what services you offer and align them to the organization.

2. Are the steps security will follow documented? Again, this helps you define what you’re doing to ensure it’s repeatable.

3. Is there a clear way to resolve disagreements? Defining an escalation path and who gets to decide what helps you get everyone on the same page around what you’re going to do to make disagreements flow professionally.

4. Can you estimate how long it will take? If you don’t know how long it’s going to take, how well do you know what it is?

5. Do you have software that helps you execute consistently? Software is a wonderful accelerator. It’s repeatable, consistent and usually fast.

6. Do you have a post-mortem process that helps you improve? Looking back often to see what went well or where the system broke is critical to any effort to improve, and it is a key identifier used to identify mature organizations.

If failures you identify in your organization often turn up in this list, the place to start is, obviously, with those issues.
As your organization moves further into the cloud, it will transform the way it delivers software and services. Those transformations are happening in development and operations organizations, and those are going to drive the transformation of security organizations. It’s not a one-time change; it’s an acceleration of change, and security has to be comfortable being a part of that. The way to find your path forward is to think about what those changes mean to you and your team.

The next critical step is to think about what changes you need to make, who you need to engage with to make them happen and what the next step is in that process. The exact answers depend on your circumstances. From one perspective, however, the conclusion will be the same: You’ll probably need to think about new skills, new tools and a different approach to organizational maturity than you did before.

The goal is for security to evolve to a point where it is an essential part of the software development process. This needs to underpin your thinking about skills, tools and collaboration across the organization. All of which means a more mature approach to security in the emerging world.
Adam Shostack is an entrepreneur, technologist, author and game designer. He’s a member of the Black Hat Review Board and helped found the CVE, among other things. Adam is currently helping a variety of organizations improve their security, as well as advising and mentoring startups as a MACH37™ Star Mentor. At Microsoft, he drove the Autorun fix into Windows Update, was the lead designer of the SDL Threat Modeling Tool v3 and created the “Elevation of Privilege” game. Adam is author of “Threat Modeling: Designing for Security,” and co-author of “The New School of Information Security.”

SANS would like to thank the sponsor of this whitepaper:

VERACODE
# Upcoming SANS App Sec Training

<table>
<thead>
<tr>
<th>Training</th>
<th>Location</th>
<th>Dates</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Univ. of California - Davis - DEV522: Defending Web Applications Security Essentials</td>
<td>Online Only, CA</td>
<td>May 11, 2020 - May 16, 2020</td>
<td>vLive</td>
</tr>
<tr>
<td>SANS Network Security 2020</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Online</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>