SEARCHSECURITY.COM

technical guide on

APPLICATION SECURITY: secure development

contents

TechTarget

SECURE DEVELOPMENT

Uneasy Union: Security and Development

Exerting security influence over development processes and organizations requires a deft touch, an advocate and the right metrics. BY CORY SCOTT

WHEN SECURITY PRACTITIONERS attempt to introduce secure development practices into a development process and organization, they often experience feelings of dizziness, nausea and vertigo. The reason is simple: trying to introduce security from the outside-in is much like trying to perform a tune-up on a vehicle going 65 miles per hour down the highway. The unpleasant feelings eventually shift to malaise and ennui as they repeatedly get brushed aside or fall off of the hood of the car.

The dirty little secret is that many development organizations are often impenetrable to outsiders for a reason. Development organizations reject formal internal structure and process imposed from the outside except begrudgingly when things have gone wrong. There's absolutely nothing wrong with this–it introduces a natural immunity to wasteful and unneeded actions and focuses on producing results for the business. And the business causes a good deal of this chaos, with frequent shifts in focus based on customer demands or changes in priorities. That is why you will often see external interfaces to the development organization, such as requirements definition and user acceptance testing, much more formally defined than internal processes.

Development organizations reject formal internal structure and process imposed from the outside except begrudgingly when things have gone wrong.

What this produces is an ad hoc (and often unspoken) risk management decision that says introducing security in the development lifecycle is not worth doing unless there is a specific risk aversion produced as a business-defined requirement. And as businesses often end up taking risks that are deemed inappropriate in some sectors, regulatory or industry compliance comes in to make the decision for them, although the response is usually a brush off with a checklist or fresh coat of paint on a structurally unsound set of practices.

So what happens when an application security incident occurs? At that point, the organization goes through a re-evaluation of the risk management decision made prior, although it is rarely couched in risk terms. Instead, it usually follows the script

SECURE DEVELOPMENT
SPONSOR RESOURCES

of a pronouncement along the lines of "We're sorry. We will make sure this never happens again!" Depending on how thorough and considerate the decision makers are, meaningful change can take place, and savvy information security managers can influence it.

THREE APPROACHES TO CONSIDER

As a result, what can we do, either in good or bad times? Going back to the speeding car metaphor, we can try to modify the car, change the rules of the road or we can change the route the road takes.

Modifying the car is most difficult, because you are trying to change the internal processes of the development organization. But with this approach, you will have the most direct influence on code quality and security. Adopting standard secure-by-default libraries and providing training for developers also falls into this category. You also have to get the most buy-in and the greatest amount of cooperation with the development team.

If you change the rules of the road, you are influencing development policy and the business requirements used to build applications. In this approach, you're working around the edges of the internal development process, trying to influence business owners, policy makers and development liaisons to use those external interfaces to your advantage to improve security.

Finally, if you change the route of the road, you're introducing technology or assessment pitstops to evaluate the car and its progress, and make course corrections as you go. You can also verify that the car has been doing what it is supposed to be doing while on the open road. It's no surprise that this approach is the most popular, because most solutions can integrate in whatever development methodologies or business processes

Secure Development Resources

WHEN STARTING to look at all of the potential application security program options, the choices can be a bit overwhelming. Luckily, there are several application security organizational models that outline common approaches and provide a decent workflow model that can be matched to your development methodology. Three models worth evaluating are the Open Software Security Assurance Model (OpenSAMM) from OWASP, the Building Security In Maturity Model (BSIMM), and the Microsoft Secure Development Lifecycle (SDL) Optimization Model.

It's impossible to talk about building security into the development lifecycle without talking about Microsoft. They were one of the first companies to share their experiences and processes in securing their products. Over a five-year period, different versions of the SDL have been released, including guidance in how to integrate Microsoft's advice through their SDL Optimization Model.

If you have a traditional development model such as the waterfall or spiral methodology for building products, the standard Microsoft SDL may be useful, while if you use Agile methodologies, the SDL-Agile requirements are more appropriate. Line-of-business applications that may benefit from something other than a product-centric approach are addressed with the Line of Business SDL. Finally, there is a "simplified SDL" that is aimed at getting started with a smaller set of requirements.› –CORY SCOTT

currently exist.

In all of these approaches, there is an element of risk management that must be addressed. "How much is enough? Or at least enough for now?" is a question that is frequently asked and very seldom answered with much depth.

General measuring sticks such as industry best practices or compliance requirements are often brought out without much definition or agreement on what they really mean. In the case of a post-incident introspective moment, one may ask "If we did these things prior to the loss, would we have been okay?"

While a risk assessment methodology for application security, either at the program level or at an individual application level, is outside the scope here, it is important to clearly define what you are trying to protect, what you're protecting it from, and some simple cost-benefit analysis.

Whether you are consciously doing it, your adoption or lack thereof of an application security program is a risk management decision. It's better to go in with your eyes wide open, even if all of the data may be hard to define. You'll find that talking to the business and development teams will also be smoother.

AN APPLICATION SECURITY ADVOCATE MUST LEAD THE WAY

Successful application security programs in development organizations vary on approach, depth, responsibilities and requirements. But they all have one thing in common: one or more people who are passionate about the problem and are able to influence their peers and management. Depending on the advocate's role in the organization, he or she may lead by example through adopting improvements in their areas of responsibility and sharing the results, or by setting the tone through mandating change in the organization.

The advocate should work with the management teams from business and development to determine a feasible application security strategy that attempts to define the goals of the program and what the timeline for the objectives will be.

A good strategy will be organization-specific and incident-aware.

A good strategy will be organization-specific and incident-aware. It is important to translate the goals into risk analysis and tradeoffs that are aligned to the organization's general risk tolerance and posture. For example, if the business is risk-averse on availability and has high redundancy and fault-tolerance requirements, engage the strategy direction on how the program will work to reduce the likelihood that a security vulnerability will disrupt operations and how the existing redundancy and fault-tolerance solutions will be of little help.

If there has been an incident in the past, a walkthrough of the root cause analysis and how the goals of the program will reduce the occurrence or probability of the root causes happening again is critical.

Another thing successful programs do is start small. Setting up a pilot program that is close to the advocate's area of responsibility is a great way to demonstrate value and

figure out what works and what doesn't. It's also not too early to start to figure out how to capture relevant metrics that can be used to measure the success of the program and how effective certain initiatives will be in reducing risk in the organization.

COLLECTING KEY RISK AND PERFORMANCE METRICS

Metrics are something that a lot of organizations struggle with, because most of the time, until you actually start the program, you don't have a good snapshot of data that is relevant to the problem area.

One thing that should help is to divide the problem into two categories. The first bucket is a set of key risk indicators (KRIs) that attempt to measure the relative risk present in the application portfolio at a given point of time. Depending on the level of application vulnerability assessment you already have, reporting on the number of open vulnerabilities per application is a good KRI. Other KRIs may include number of externally reported vulnerabilities per application, number of unassessed applications, and number of application security incidents in a given time period. The second bucket is a set of key performance indicators (KPIs) for the program that measures a program's effectiveness and efficiency. Number of applications assessed or impacted, number of vulnerabilities identified, number of vulnerabilities remediated, and other coverage measurements are useful KPIs.

Depending on the level of application vulnerability assessment you already have, reporting on the number of open vulnerabilities per application is a good KRI.

If you introduce development initiatives that attempt to build security in, then capturing the percentage of applications using secure libraries or established architectures or percentage of developers trained would also be a good idea.

WAYS TO BUILD IN SECURITY

One trend many organizations have started to pursue is the use of "secure-by-default" libraries or application framework settings that handle tricky issues such as input validation and output encoding, along with common cryptographic mistakes.

OWASP provides a well-known library called ESAPI that has garnered attention from the development community. Also, modern Web frameworks are starting to include common sanitization libraries that can eliminate the need for custom validation routines.

From an internal development perspective, architecture teams have found value in extending security requirements and peer and expert-reviewed designs to application architecture specifications and design patterns.

Developer training in secure coding practices and basic application security principles is a common component of application security programs. After all, developers don't intentionally set out to write vulnerable code at the start of every workday. However, it is important to make sure that the training is engaging, relevant and respectful of schedules.

Training can be engaging when it is hands-on and interactive. Look for training providers that come to your location and have lots of lab exercises interspersed with key concept lectures. Training curriculum should also be relevant, which means it should use your developer's code and previous vulnerability histories. It is worth the investment to do an assessment that includes code review prior to training the developers who wrote the code. It will give the training provider guidance on what is actually being used by the development team and what issues they are struggling with. Finally, it needs to be consumable in a way that doesn't impact development schedules—try to rotate small groups through on-site training if possible.

Getting feedback from the training is also important. Test the development team on what they've learned, and also get their impression of how effective the training was. Successful training often will result in developers leaving the class and searching their own source for where they may have made mistakes, so make sure you conduct a survey after they've had a chance to do so.

HOW TO BEST ANALYZE SOURCE CODE

Static code analysis is a common process used by many development and security organizations to find flaws in code prior to release. At first, code review was primarily a manual process. These days, automation is significantly more common.

Static code analysis tools will search through source code for common programming errors, attempt to identify sources of untrusted—also known as tainted—input, and in some cases, generate an intermediate compiled object to evaluate the execution flow to determine if a given application, when compiled with known frameworks, is vulnerable to anomalous or malicious input.

The most successful use of static code analysis is a strong combination of manual review expertise brought by personnel mixed with the capacity and coverage that automated tools provide.

The most successful use of static code analysis is a strong combination of manual review expertise brought by personnel mixed with the capacity and coverage that automated tools provide.

If you adopt the use of automated tools without a clear definition of what you are looking for or what is important to you, you risk getting overwhelmed with findings that require significant investigation and validation against a non-existent standard.

If there is not expertise in security code review within your organization, consider the restricted use of static code analysis tools to the vulnerability classes that have the greatest impact in your application's risk profile and that are well understood by the development team. In initial rollouts of static code analysis technologies, try to team a developer with build expertise with a security assessment specialist who has code review experience to determine how effective the toolchain will be and how much tuning will be required before it is rolled out to a wider audience.

When you're ready to build your application security program, it's easy to become

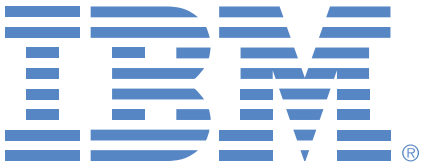
overwhelmed with the challenges of modifying process, introducing new technology, and getting people to change how they approach potential risks introduced by application development. If you clearly outline how the business is impacted by vulnerabilities in the application portfolio and either find an advocate or become one yourself, you can start to map out an approach that makes sense to your organization. Starting small, collecting metrics as you go, and using established process and technology frameworks will increase your chances of success.›

Cory Scott is the regional director for consulting services at Matasano Security.

TECHTARGET SECURITY MEDIA GROUP

TechTarget
Where Serious Technology Buyers Decide

"Technical Guide on Application Security: Secure Development" is published by TechTarget, 275 Grove Street, Newton, MA 02466 U.S.A.; Toll-Free 888-274-4111; Phone 617-431-9200; Fax 617-431-9201.



- **Effectively design security into your products and services in a way resilient to change**
- **Business case for data protection: A Study of CEOs and other C-level Executives**
- **Securing today's applications: Design, deliver and secure smarter software and services**

About IBM:
At IBM, we strive to lead in the creation, development and manufacture of the industry's most advanced information technologies, including computer systems, software, networking systems, storage devices and microelectronics. We translate these advanced technologies into value for our customers through our professional solutions and services businesses worldwide.