WHITE PAPER

Agile Security

Successful Application Security Testing for Agile Development

VERACODE

Software Security Simplified

Abstract

It is an imperative to include security testing in application development. Yet, with Agile's fast pace, and lean concepts, it easy to see how many organizations would simply consider testing for application security defects to be too costly in terms of both time and resources. The reasons behind these beliefs are concerns over the cost of the tooling versus the benefit, the cost of deployment and training of the tools, the inability for these tools to fit into Agile development processes, and the objections of developers who must become proficient in the use of the tools. This paper addresses these concerns and describes methods that utilize Veracode's Security Review and methodologies for security testing that succeed in the Agile world.

Introduction

As competitive and economic pressures cause business cycles to shorten, the need to deliver quality software at a much faster rate increases. Over the past decade this pressure has been the force behind the shift from the traditional 'waterfall' development approach (with its sequential, orderly phases and milestones) to the Agile development methodology, utilizing eXtreme Programming (XP), Scrum, or other project management and development methods associated with the "Agile Development Movement". These patterns and practices seek to implement the principles placed forward in the Agile Manifesto, which defines software development practices that promote highly iterative development, open collaboration, "good- enough" design and documentation, lean and minimal process and tools and fearless process adaptability throughout the product lifecycle.

The strength of Agile is that it can save organizations significant amounts of development time and money, while still allowing them to deliver high-quality software.

There is the perception today that these Agile methods do not embrace secure code and coding practices, and to some extent, historically, security has not been given the attention it needs when developing software with Agile methods. With Agile's fast pace, it's easy to see how many organizations would simply consider testing for application security defects to be too costly in terms of both time and resources.

But in reality, since Agile methods focus on rapidly creating features that satisfy the customers' needs, and security is a customer need, it is important that it not be overlooked. In today's highly interconnected world, where there are strong regulatory and privacy requirements to protect private data, security must be treated as a high customer priority.

Additionally, the financial cost of not including security testing within an Agile, or any, methodology can far outweigh the short-term benefits of not including it. Producing insecure software dramatically raise the costs of correction. According to the National Institute of Standards and Technology (NIST), it is 6.5 times more expensive to find and fix any flaw in development than during design, 15 times more in testing, and 100 times more in production. The cost to find security flaws using traditional testing is even much higher since you are not testing what the application does but what it unintentionally allows.

The real cost of insecure software today comes from data loss, process interruption, and brand damage from criminal attacks, and they are far more consequential today than ever. The danger today is not only a single incursion that steals data, but a single incursion where malware is left behind and thefts can continue across a much larger period of time or a process can be interrupted at the attacker's will. Leaving one hole in only one application can quickly result in disaster. Lack of application security testing also jeopardizes compliance with federal regulations such as HIPPA, PCI, and Sarbanes-Oxley (each of which has hefty fines for electronic data breaches).

It's obvious that application security cannot be ignored, even with the adoption of Agile development practices. It is an imperative to include security testing as part of your development.

With the knowledge that you have real costs and a customer imperative for the inclusion of application security testing in your delivery, how do you incorporate it into your Agile methodology? More importantly, how can you accomplish it without incurring more costs than you are attempting to save with the adoption of an Agile development process, and how do you keep from grinding deliveries to a halt on what is supposed to be 'Agile' development?

Staying Lean: The real cost of ownership in an Agile world

The issues that are encountered in the adoption of application security testing in an Agile approach, if not understood, can cause increased time and cost, anathemas to very essence of what Agile proposed to do for organizations. This is discovered from the start in the selection of the application security analysis capability for the organization. There are several to choose from and while all can provide benefit, they have to be rated in how they fit in terms of what the organization is attempting to do by being 'Agile'. Anything that blocks progress or slows down development cycles is counter to the principles of Agile development. Additionally, time and people cost money, so the selection of application security analysis must include what the cost of ownership will be - the calculation of the total cost in terms of money, time and people to acquire, implement, train, rollout throughout the enterprise, and maintain this capability within the organization.

The first decision point will be whether or not to do anything in-house, or outsource the security testing to independent third parties who will perform manual penetration testing. While these are valuable services and can be helpful in an overall security plan, outsourced testing requires too much time to be feasible in Agile development where entire development cycles and releases are measured in weeks.

In-house tools have their benefits, but most of these tools carry with them unforeseen costs. Use of an in-house tool incurs total cost of ownership issues that has to be considered. Many of these tools involve protracted amounts of time to install, configure the tooling to get it to a point of returning actionable security flaws back to the development team. The tools become yet another tool to add into the tool chain to be configured and run, at a time when the organization is attempting to be lean. Most of these in-house tools require developer training in how to use the tools, and even much more training in how to configure the tool to provide results back that are not 'noisy'—results that are full of expected defects that are not really defects, called false positives.

Most in-house security testing tools focus on the analysis of the available source code, which is yet another limitation that can increase overall cost. The world of application development today includes code acquired from many sources—open source, shared and purchased. This third-party software of unknown pedigree (sometimes referred to as SOUP) cannot be fully tested in the applications using source code analysis tools and leaves areas of the application open to potential security flaws.

Finally, these tools require training for the administration of the tool, adding to the responsibilities of the development organization that can add to resource loading or inefficiency. The costs of installation and deployment of the tool sadly does not diminish with scale. The installation and setup and care and feeding of the tool is an additional cost in time and people for each new application team throughout the enterprise that wishes to adopt application security analysis.

There is another way—one that combines the value of independent verification and integration with Agile development practices but without the deficiencies and excessive cost of ownership of in-house tools. This is the Veracode solution: A 'cloud' based service that is invoked when the Agile team wishes and that can be integrated with the IDEs, defect tracking, and build systems the team has chosen. In this approach there is no installation, no setup,

no tool training for developers, no learning curve to determine false positives. There is no 'tool' to incorporate into a build cycle, no lengthy in-house analysis to wait for that slows down the Agile process. The Veracode approach performs static and dynamic analysis on the binary application that is built and uploaded into its secure cloud hosted platform and is analyzed and returned with the most accurate results in the industry, usually within 24 hours. The cloud based approach also allows the addition of application security testing without the need to add additional hardware to provide for the extra processing power needed to perform the analysis in-house.

Accuracy is especially worth emphasizing as a key component to success in Agile development. "Noisy" assessments cause developers to waste time in the determination of what is actually a real defect, limiting what can be accomplished in their development cycle.

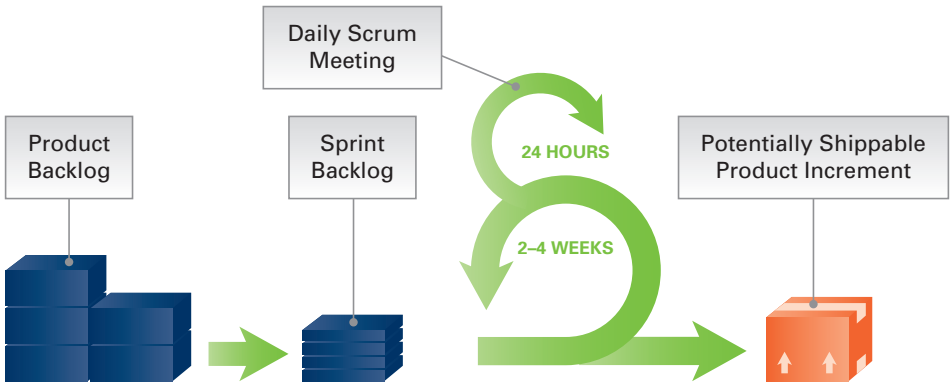
Veracode's accuracy comes from two areas. The first is Veracode's patented static binary analysis technology that verifies the final integrated application i.e. the binary or bytecode. The Agile team simply uploads the executables to the Veracode platform and the analysis is performed in the cloud. This form of security testing represents the most accurate form of security testing available. Binary or bytecode is the truest representation of final application. It is what the machine executes and what gets attacked, not the source code. It can test all the code, even the SOUP code from the included open source and other components that were not available for source code checking. By verifying the application code at the same level that it is attacked, Veracode's static binary analysis technology ensures that critical threats and vulnerabilities are detected. The second benefit of the static binary analysis is that the assessment of the application in its final form also allows for lower false negative and false positive rates than alternatives. Veracode's false positive rate is often 100% lower than source code scanning technologies that tend to be "noisy". Additionally, the analysis and assessment is performed with the latest and best data on the identity and discovery of attacks and vulnerabilities that the industry has to offer. The cloud-based solution always is providing the most up to date version of the analysis, as opposed to an installed tool that may be several or many months old between versions of delivered updates from the vendor.

Staying Agile: Security analysis in the Agile process

The rapid acquisition and deployment of a capability such as Veracode's is only truly beneficial if it can become part of the organizations' defined Agile process. Any tool that requires the development team to dramatically modify their process defeats the lean and Agile goal of the team. The key to success is the in the seamless integration of application security into an Agile, or any defined, development process. There are several methods defined that help organizations adopt an Agile approach to development. In our examples we will use Scrum.

Scrum (sometimes seen spelled as SCRUM) has been used to develop complex products since the early 1990s. Scrum is grounded in empirical process control theory, and employs an iterative, incremental approach to optimize predictability and control risk. A Scrum approach consists of a Team or set of Teams designed to optimize flexibility and productivity; to this end, they are self-organizing, they are cross-functional, and they work in iterations. The Team consists of developers with all the skills to turn the product requirements into a potentially releasable piece of the product.

Scrum teams work in iterations that are time-boxed: Scrum employs time boxes to create regularity.



The heart of Scrum is a Sprint, which is an iteration of usually a 2-6 week period of time that is consistent throughout a development effort. All Sprints use the same approach, and all Sprints deliver an increment of the final product that is potentially releasable.

The adoption of application security analysis and assessment usually occurs in one of two ways in a Scrum approach: the Security Sprint or the Every-Sprint approach.

The Security Sprint approach

The security Sprint is a common approach for the inclusion of application security into the agile Scrum model. Organizations such as The Open Web Application Security Project (OWASP) has identified and discussed the concept of Security Sprints as part of its recommended approaches to application security.

In the Security Sprint approach, periodic Sprints are performed focused exclusively on security. As with any Sprint, they are the implementation of User generated stories that drive development. In a Security Sprint the stories are developed and implemented just like any other stories allowing the other steps in a Sprints such as Test-Driven Development, Unit test, Continuous reviews and integration are all the same: the only difference is that the implemented stories are security related. Examples of the type of stories that are implemented in a Security Sprint would include Intrusion Detection, Logging, Authentication and Authorization, and Error Handling. Some technical risks, such as XSS, also need to be identified as a Story.

Security Sprint

In the Security Sprint approach with Veracode, the team would utilize Veracode's Security Review at the start of the Sprint, with the current build and upload of that build to the cloud-based analysis and assessment. The returned assessment is then used as the basis of the backlog for that Sprint. Since the analysis returned likely contains more identified defects and vulnerabilities than were identified as the user stories implemented and supported by the Sprint, the Sprint identified defects become the essential Sprint backlog work items for the Sprint. If there are too many identified, then the cross-functional team must prioritize which stories are to implemented and corrected, with the rest put into Product back log for another Sprint.

The Every-Sprint approach

The other Sprint approach is referred as the 'every-Sprint approach'. This is the recommended approach for the adaptation of agile to methodologies such as the Microsoft Security Development Lifecycle (SDL). It is also the method that is utilized by the Veracode Development team in their implementation of Scrum used in the delivery of Veracode SecurityReview®.

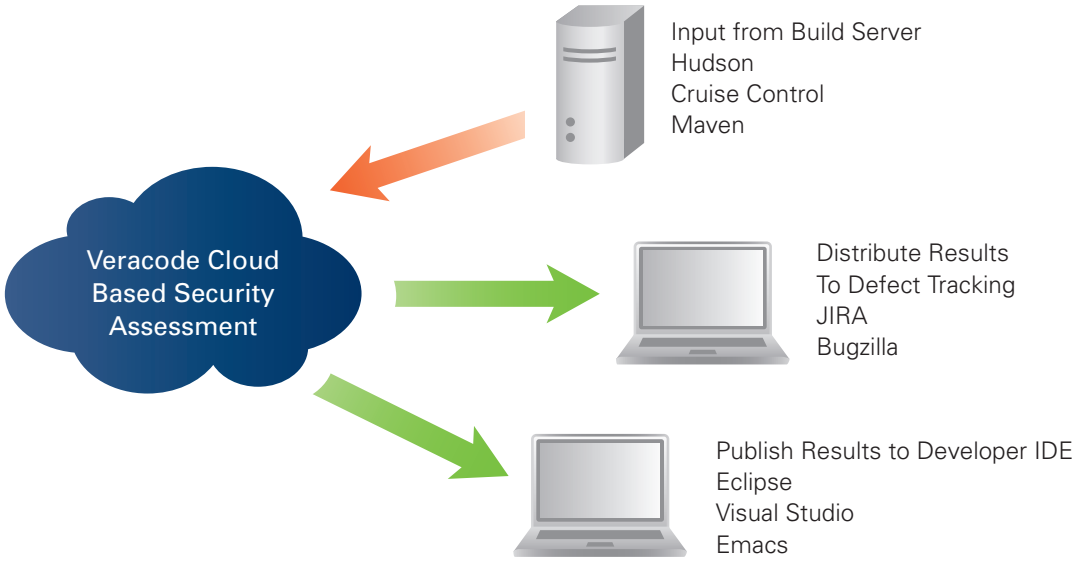
In the every-Sprint approach, security requirements and user stories are categorized by frequency of completion. The first category consists of the requirements and stories that are so essential to security that no software should ever be released without these requirements being met. Whether a team's Sprint is two weeks or two months long, every security requirement in the every-Sprint category must be completed in each and every Sprint, or the Sprint is deemed incomplete, and the software cannot be released. An example of this, from the Veracode engineering approach, is that XSS detection, along with any defect detected from the PCI/SANS top 25 /OWASP top 10 is considered an every-Sprint must fix requirement. The remaining category consists of requirements or stories that are not so critical as to be mandated for each Sprint. This category is called the bucket category. The key is to establish the every-Sprint and bucket rankings.

Once the rankings are established then they are applied every Sprint. In the Veracode Sprint, which is 6 weeks in duration, the first three weeks are focused on coding, with a code complete at the end of the third week. At this junction, the new baseline applications are uploaded and analyzed. The returned analysis result is considered part of the three-week testing cycle, and vulnerabilities and defects identified are corrected and tested as part of the Sprint. Bucket category issues are worked on a time basis.

Staying Relevant: Agile Developers need Agile tools

The Agile manifesto declares: "Individuals and interactions over processes and tools." This statement doesn't mean you can forget your tools. It's really about making sure that any tool brought in helps facilitate the interactions between team members. The problem is that many of the current security tools in the market today have been designed by and for people with traditional development models in mind. Many actually hinder the process by following a "waterfall" like model of isolating code and code changes so they can analyze them. The Scrum teams should be able to function together and in parallel, while integrating work together on a regular basis.

Whether you're moving through a 6-week Scrum Sprint, or using continuous integration and automated tests your security tools should support the model and other developer tools that make your team most productive. The addition of application security testing should not require your team to change how they interact with their code, how they identify and work defects or how they interact with software configuration management (SCM) tools. Whether the defined tool chain is Microsoft Visual Studio based, using Team Server, or Java tooling using Eclipse as the IDE , Subversion and JIRA, or if the development team is using Bugzilla and Emacs, the addition of an application security assessment tool should not require developers to use yet another IDE to work the defects, or change how they work day to day – that would defeat many of the agile principles for being fast , lean and productive. Many of the security tools available in the market today require special, unique additional IDE environments that developers must use to view the identified defects, and use yet another additional interface to track the progress of the fixes. Developers do not want yet another tool to learn and use, and most want as few tools as possible to get their job accomplished. Many see additional tools as a hindrance.



The Veracode approach is designed to be tool chain agnostic, open, and not to require any changes to the tool chain. Veracode's cloud based analysis provides API integrations into the popular tools used by agile teams, from Emacs, Visual Studio and Eclipse IDE, Build systems such as Hudson and Cruise Control, and Defect and Change Management tools such as JIRA and Bugzilla.

The integration points to the cloud are both input and output. Integration into the Agile teams' build system such as Hudson allows the newly created binary to be uploaded to the Veracode cloud based system where it is analyzed. The results of the analysis can then be published back out to the team, into existing Defect Tracking systems such as JIRA and Bugzilla, and also into what the developers IDE of choice is, including Visual Studio and Eclipse. The strength of this approach removes any 'impedance mismatch' between tools that can slow the progress or impede the interaction between team members. The team uses the tool chain it has already been comfortable using.

All of these steps and decisions lead up to what you are trying to enable—Agile Developers finding, fixing and limiting security problems in their code and coding practices. But attacks continuously evolve—problems like Cross Site Scripting (XSS), Insecure Direct Object Reference, Broken Authentication and Session Management, Injection Flaws, Cross Site Request Forgery (CSRF), Failure to Restrict URL Access, Malicious File Execution and Information Leakage and Improper Error Handling were not even thought about several years ago, and new attacks are constantly surfacing. Developers must understand the attacks and controls to properly mitigate the threats, because ultimately, everyone on the team responsible for security. Therefore, all developers should have training and an understanding of application security. If an Agile developer's precious training time is dedicated to becoming versed in using a tool and understanding how to wade through the data to find the flaws and not waste time on the false positives, they are not becoming security experts or secure developers. Secure developers come from being certified in secure development practices and writing secure code in the first place, which is what Veracode eLearning service provides. Developers are provided with a path to understanding how NOT to keep making the same mistake from Sprint to Sprint and helps the development organization establish secure coding and coding practices.

Summary

Secure software comes from independently verifying final, integrated applications created by secure—or insecure—development as completely, accurately, and affordably as possible. The perception today that Agile methods cannot or do not embrace secure code and coding practices and application security testing is, in fact, false. The Veracode cloud based application security analysis is not only capable, but actually superior in its incorporation and use in Agile development. The objections and concerns raised concerning the total cost of acquisition and use of traditional in-house tools are mitigated by the cloud-based solution. The concern over the cost of additional human resource with added training and assessment time is mitigated through the unparalleled accuracy of the Veracode binary analysis solution. The concern that an application security tool cannot fit into an Agile process has been shown to be unfounded as presented by the examples of types of Scrum Sprints that actually utilize Veracode SecurityReview. Any objections from developers over the use of new and additional interfaces or tools are mitigated through the open extensible API integration capability provided by Veracode SecurityReview. Finally, developer training is focused on becoming a certified secure developer not a tool expert.

WHITE PAPER

VERACODE

Software Security Simplified

Veracode, Inc.
4 Van de Graaff Drive
Burlington, MA 01803

Tel +1.781.425.6040
Fax +1.781.425.6039

www.veracode.com

WP/AS/0610

ABOUT VERACODE

Veracode is the world's leader in cloud-based application risk management. Veracode SecurityReview is the industry's first solution to use patented binary code analysis, dynamic web assessments, and partner or Veracode delivered manual penetration testing, combined with developer e-learning and access to open source security ratings to independently assess and manage application risk across internally developed applications and third-party software without exposing a company's source code. Delivered as a cloud-based service, Veracode provides the simplest, most complete, and most accurate way to implement security best practices, reduce operational cost and comply with internal security policies or external standards such as OWASP Top 10, CWE/SANS Top 25 and PCI.