APPSEC DEPLOYMENT USE CASES

BY VERACODE

VERACODE

APPSEC DEPLOYMENT USE CASES

THERE'S MORE THAN ONE WAY TO DO APPLICATION SECURITY

Application security is like jazz: You can play it with any number of instruments, and there are a large number of combinations. Your decisions will depend on the programming language, the type of application (web, mobile, desktop, microservice), your development cadence, your development tool chain, your security stack, your level of acceptable risk (by application), and the AppSec vendor you choose. As with a jazz band, it takes some experience and skill to pick the right combination for your task. At Veracode, we're here to help you with these decisions.

This document aims to show you different deployment use cases to give you some insights into how Veracode typically integrates into the development processes. We'll walk through several deployment use cases at VeraBank Inc, a fictitious company, to discuss different deployment options.

FINDING THE RIGHT MIX OF ASSESSMENT TYPES

The most common ways to assess application security are:

- **Static Analysis:** Assessing the security of an application's first-party code, in other words, scanning the binaries or the source code of the code you have written yourself
- **Software Composition Analysis:** Assessing the security of an application's open source code by highlighting known vulnerabilities in these components
- **Dynamic Analysis:** Assessing the security of a web application by interacting with it and trying to exploit it
- **Manual Penetration Testing:** Assessing the security of an application through an ethical hacker who tries to exploit the application

Unfortunately, you can't just pick one of these assessment types and be done with it because they find different things. The following table provides a high-level overview:

Capabilities	Static Analysis	Software Composition Analysis	Dynamic Analysis	Manual Penetration Testing
Flaws in Custom Web Apps (CWEs)	✓		✓	✓
Flaws in Custom non-Web Apps (CWEs)	✓			✓
Flaws in Custom Mobile Apps (CWEs)	✓			✓
Known Vulnerabilities in Open Source Components (CVEs)		✓		✓[1]
Behavioral Issues (CWEs)	✓[2]			✓
Configuration Errors (CWEs)			✓	✓

[1] Penetration testing can find known vulnerabilities in open source components but this may not be as rigorous as Veracode Software Composition Analysis, which not only systematically flags CVEs but also crawls commit histories and bug tracking tickets in open source projects to identify silent fixes of security issues.

[2] This is not true for all static analyzers. Veracode can exercise the code and manipulate the UI for behavioral analysis in mobile applications.

Business Logic Flaws (CWEs)				✓
Can be automated (scalability, cost, speed)	✓	✓	✓	

For more details on this subject, please read the Veracode white paper Your Guide to Application Security Solutions.

Background Knowledge

Synchronous vs. Asynchronous Scanning

Most teams think of scanning as a gating factor before release. We call this **synchronous scanning**, and it's a good idea to do this if you don't release more than once a day and haven't implemented any other ways to scan your code earlier in the SDLC.

However, some teams release several times a day so that some scan types, for example a full static analysis policy scan or even a dynamic analysis scan, take longer than you have time for. On average, most of our static analysis scans complete in less than 15 minutes, while 90% of dynamic analysis scans complete in less than 24 hours (they have to scan at a reasonable speed so they don't crash your web application). In these cases, we recommend that you use Veracode Greenlight to test your code earlier on, either in the IDE or with every check-in. Veracode Greenlight scans only the file you're working on, so it only triggers on flaws that are fully contained in that file and where the source and sink don't spread over several files. This still catches a very good amount of flaws. In addition, if you release very often, you are only making very small changes to the code each time, which reduce your risk of introducing a new flaw. In these cases, it is often an acceptable level of risk to release to production after Veracode Greenlight has completed (which takes seconds), to run the longer scans overnight and to address any flaws that may have escaped to production in the morning. We call this **asynchronous scanning**.

Whether you're leaning towards speed or deep scanning ultimately depends on your development style and risk acceptance, but it's good to keep both models in mind when assessing the best way to introduce application security to a development team.

WHEN TO USE WHICH ASSESSMENT TYPE

Veracode offers a breadth of application security assessment types. In this section, we outline where to apply which technology, ***independently of whether these are from Veracode or an alternative source***.

Assessment Type	Advantages	Limitations
Static Analysis (with Entire Application in Scope)	▪ Very broad coverage of flaw types (CWEs) ▪ Looks at the flaws in the context of the entire application, analyzing all the data paths. ▪ Can scan any type of application, including web, mobile, desktop, or microservices ▪ Scanning frequency should be in line with how often developers can review scan results ▪ Use static analysis as part of Continuous Delivery pipeline and file security issues in bug tracking system	▪ Does not provide instant feedback to developers as they're coding. ▪ Cannot find CWEs related to server configurations ▪ Limited to code that developers can remediate. Does not report vulnerabilities in 3rd party components (see: SCA).

	▪ Can track flaw history: new, open, fixed. Important for trending reports on mean time to remediation. ▪ Suitable for compliance purposes	
Static Analysis (on File Level)	▪ Recommended for development teams who want to shift left in application security testing by scanning early and often. Scans usually complete in seconds. ▪ Best suited when scanning multiple times a day ▪ Recommended for use by developers working on the new code for continuous flaw feedback and remediation guidance ▪ Developer friendliness: enhances learning, allows developers to find and address issues without exposing flaws in reports.	▪ Scans individual files, so can only detect vulnerabilities where source and sink are in same file ▪ Typically not suited for compliance scanning because scope limitations may cause false negatives ▪ Does not report vulnerabilities in 3rd party components
Dynamic Analysis	▪ Scans web applications without having to integrate with the SDLC ▪ Ability to scan in pre-production and production ▪ Suitable for compliance purposes	▪ Scan times are often between 12 and 24 hours for complex applications, so recommended for overnight scans, or for asynchronous scanning
Software Composition Analysis	▪ Finds vulnerabilities in 3rd party components ▪ Scans take seconds or minutes ▪ Can scan any type of application, including web, mobile, desktop, or microservices ▪ Suitable for compliance purposes	▪ Does not find flaws in first-party code
Manual Penetration Testing	▪ Works on all types of applications and languages ▪ Only assessment type to find business logic flaws that require understanding of the application's business functionality and impact ▪ Suitable for compliance purposes	▪ Slow, human-driven process, typically about five to 10 days per application

GENERAL VERACODE SCANNING RECOMMENDATIONS

The following recommendations are specific to Veracode solutions:

- **Policy scans**: These should represent the code deployed to production. With Veracode Static Analysis, any code that gets scanned that *isn't* in production should be scanned in a developer sandbox. For Veracode Dynamic Analysis, there is no concept of a sandbox, but you can conduct policy scans for the pre-production URL in a different application profile from the production URL to keep them separate. For Veracode Software Composition Analysis, you can scan every commit and simply compare the hashes values of the binaries to the software deployed in production.
- **Sandbox scan promotion**: This happens in Veracode Static Analysis when the developer sandbox scan is clean, i.e., no policy affecting flaws remaining or unmitigated. It must happen as close to release as possible to satisfy the point above. This could be as part of the release process. If the release gets cancelled, the promoted scan must be deleted. That might mean deleted in Veracode, **and** in any external reporting tool.

- **Periodic policy scanning**: Code that has been deployed into production should be scanned in a policy scan on a regular basis, regardless of whether there has been a code change. This ensures that Veracode customers benefit from enhancements to the scanning technology. Assuming you have not touched the code, anything between monthly and annual scanning of the application is adequate, depending on the criticality of the application.
- **Scanning every commit**: For scanning on every commit, we generally recommend using Veracode Greenlight, which provides a faster, light scan of your code. We generally don't recommend scanning every commit with Veracode Static Analysis, Veracode Dynamic Analysis, and Veracode Software Composition Analysis. If you are releasing more than daily, we recommend scanning nightly so scans don't overlap and cause errors in the pipeline due to existing scans running.
- **Pro Tip**: Use the Commit ID and the CI Job ID in the scan name, makes it easier to track when it has been scanned and what caused it to be scanned.

CASE STUDY: VERABANK, INC.

VeraBank, Inc. is a fictitious global bank whose investment strategy is primarily driven by investing in the booming application security testing market but they also do retail banking and commodities trading. Founded in the 1970s, they have a wide range of technologies, all the way from microservices written in Go and mobile applications written in Xamarin to COBOL mainframe applications. Most of their development is in-house, using a wide range of technologies for software development, driven partially by individual team preferences and necessities and partially by technologies brought in through acquisitions. VeraBank chose Veracode for its application security strategy and has been using its solutions for over a year now.

VERABANK INFOSEC SYSTEMS AND PROGRAMS

The VeraBank security team wants to give their software engineering teams as much leeway as possible to account for their specific needs and technologies. However, they also need to keep an overview of the entire application security landscape for the company. Their CISO demands monthly updates and briefs the board on a quarterly basis about progress on security and compliance. They also need to provide evidence of compliance to external auditors, e.g., for PCI DSS.

Because not all applications require the same risk level, VeraBank has classified each application's appropriate risk level and created custom policies for acceptable risk in the Veracode Platform.

VeraBank uses **Splunk** to track their governance, risk, and compliance progress. The **Veracode Platform** integrates with Splunk to report which applications are in and out of compliance. This helps provide a clear picture across static, dynamic, and software composition analysis, as well as manual penetration testing results, in one view.

VeraBank also uses Splunk as their SIEM to help analyze risks and threats. They have integrated Splunk with the Veracode Platform to help track vulnerabilities in their applications and elevate the threat level when they see an exploit for that application. VeraBank has also set up some dashboards to track high-level vulnerability data in Splunk to observe trending.

Because the VeraBank InfoSec team doesn't have the bandwidth for program managing the engineering teams, they have signed up for **Veracode Security Program Management** (SPM). When they first started up their AppSec program, their SPM was very helpful in categorizing their applications by risk level and defining appropriate policies. Now, they have a scheduled call with their SPM every month to go over trends in the program and which engineering teams to onboard next. Between check-in calls, the SPM works with VeraBank's engineering teams to get them onboarded into the Veracode Platform, get their first scans running, and assists with integrations with the SDLC.

Because VeraBank operates globally with a distributed IT team, it has been tough to even get a handle on all of the applications present inside the company. VeraBank uses **Veracode Discovery** to scan the Internet for unknown VeraBank websites and then leverages **Veracode Dynamic Analysis** to conduct a security scan of each application to see if there are any vulnerabilities that put the company at risk. This has helped VeraBank to identify unknown applications, which are then onboarded into the official VeraBank AppSec Program. They even identified some

websites that were risky, but outdated and no longer needed, so the bank shut them down, saving operational cost in the process.

VeraBank uses the **Imperva Web Application Firewall** (WAF) in front of their most critical applications for compliance reasons and to fend off application-level attacks. Whenever they find a vulnerability in their public-facing web apps that they cannot fix immediately, they use the integration with Veracode Dynamic Analysis to create a virtual patch for the application until the code can be fixed and deployed.

PROVIDING APPSEC KNOWLEDGE ACROSS ENGINEERING AND SECURITY TEAMS

Although AppSec is an important topic for VeraBank, most of their developers had never received secure coding training, either on the job or as part of their computer science degree. VeraBank started out by licensing **Veracode eLearning** for all of their developers, which gave them a baseline training for secure coding. This brought up the level of awareness for AppSec across the organization, helped VeraBank comply with PCI DSS requirements, and added an important item to many developers' resumes.

After the first year, VeraBank's security team saw that cross-site scripting and SQL injection attacks were still very prevalent in their code, so they organized a custom training to help educate their engineering force. This was delivered as the **Veracode Application Security Workshop** for engineers at their headquarters and as **Veracode Instructor-led Training** via WebEx for remote employees.

While VeraBank has some AppSec expertise in-house, they couldn't scale to meet the increasing demand from their engineering teams to assist with code fixes or to review mitigations for their auditors. They signed up for **Veracode Application Security Consulting** so that their developers could schedule developer consultations with an AppSec expert. In these 45-minute sessions, the developer shares the security findings and optionally their source code snippet via WebEx to get suggestions on how to best fix a vulnerability.

In some cases, when flaws cannot be fixed directly but where compensating controls mitigate the exploitability of the flaw, software engineers can submit a mitigation for a flaw. The challenge is that these should be reviewed by an independent, skilled person so they stand up to audits. VeraBank could not provide a timely turnaround for mitigations through their internal security team, so they signed up for the **Veracode Mitigation Proposal Review** package, which outsources this process and provides a clean audit trail.

USE CASE 1: ONLINE TRADING PLATFORM ON MICROSOFT .NET MONOLITH IN AZURE, BEING RE-ARCHITECTED TO MICROSERVICES

VeraBank developed its own online securities trading platform, specialized in trading a crypto currency called VeraCoin, which is backed by unicorn horns – a rare and financially stable commodity. The trading platform is based on **.NET** technology, coded in **C#,** and hosted in Microsoft's Azure cloud environment. The team is restructuring its monolith into microservices, and they have decided to run a two-pronged approach: Fix only critical security issues in the monolith, but ensure that they don't introduce new vulnerabilities in the new microservices code.

They release their new microservices code several times a day, with a focus on speed of innovation to help them attract and retain customers in the hot crypto currency market. They run a pure Microsoft stack: **Microsoft Visual Studio** is their IDE. They use cloud-based **Microsoft Azure DevOps** (formerly known as Microsoft Visual Studio Team Services) to file bug tickets, store their code, and deploy it directly to **Microsoft Azure**.

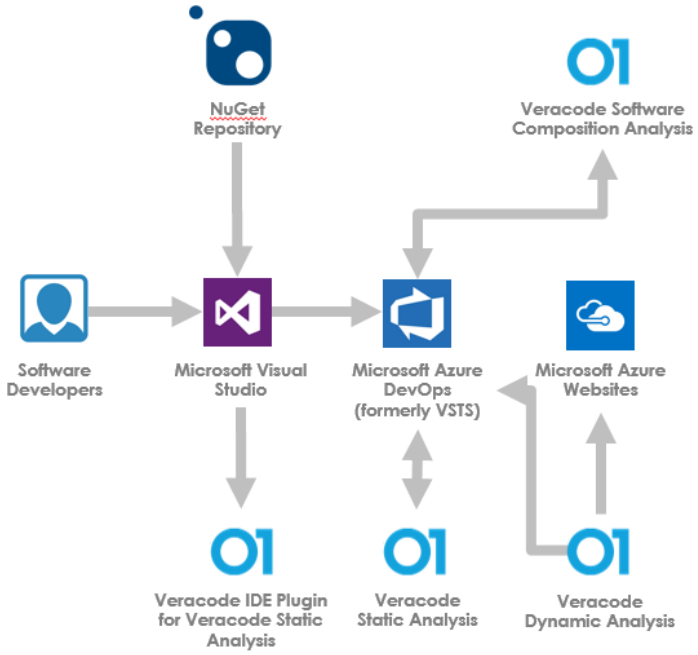
While using the same type of SDLC tooling for both microservices and monolith, their monolithic code is released only once a month.

Static Analysis in General	VeraBank uses **Veracode Static Analysis** to scan all parts of the application, both the monolith and the microservices, but the monolith and the microservices are set up in separate application profiles on the Veracode Platform because they are coded by different teams and VeraBank wants to apply different types of profiles to them. The monolith will not be supported for much longer, so VeraBank decided to only

retroactively fix critical and high vulnerabilities; the policy was adjusted accordingly. The microservices, which are new developments, have a stricter policy that also fails a build on medium vulnerabilities.

Static Analysis on Monolith

For the team maintaining the monolithic code, they upload their compiled application to the Veracode platform, and start a static analysis scan using the **Veracode Visual Studio Plugin**. The scan is performed in a **developer sandbox**, a feature of **Veracode Static Analysis** that enables engineering teams to scan code privately without promoting the scan result to Splunk. When the policy-affecting issues found in the sandbox scan have been resolved, and all mitigations have been approved, they promote the sandbox scan to a policy scan. This action updates the version and status in Splunk, and the software is then released to Microsoft Azure. Sandbox scans occur before each feature branch is merged back into the main branch; policy scans occur once a month before each release. This process helps keeping up the speed of innovation by moving functionality into production while ensuring that flaws that haven't been caught by other processes are flagged to security in less than a day.



Use Case 1 - Monolith: Monolith with a lot of technical debt that is scanned less than once a day. Developers use IDE plugin to scan in the Developer Sandbox without affecting policies, then do policy scan before release. Microsoft Azure DevOps acts as source code repository, build system, and ticketing system. Open source code is scanned with Veracode Software Composition Analysis. Veracode Dynamic Analysis is run against the production web server.

Static Analysis on Microservices

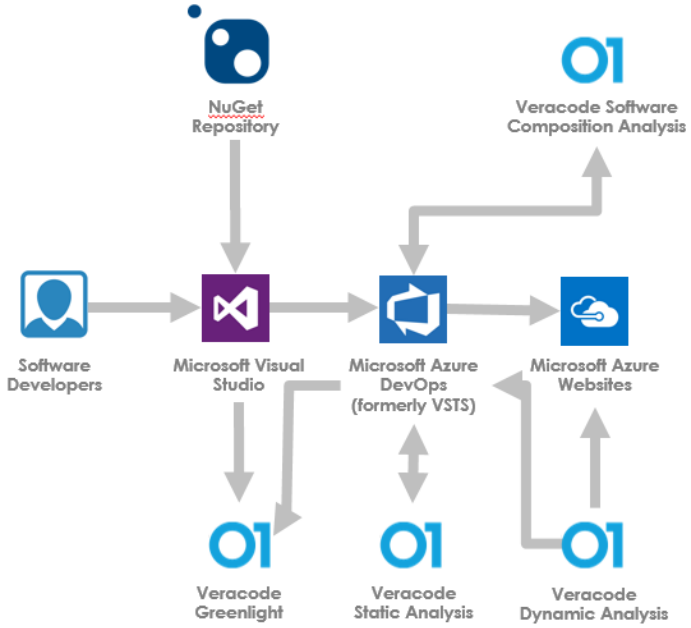
To help keep up the development velocity, the microservices teams also use **Veracode Greenlight**, which plugs into their Microsoft Visual Studio environment and continuously finds newly introduced vulnerabilities as developers are writing code. This greatly reduces the overhead for fixing vulnerabilities because developers can fix the code right then and there. This has greatly increased the pass rate of the final policy scans with Veracode Static Analysis. [Note that the monolith team decided not to use

Veracode Greenlight because it would highlight all legacy vulnerabilities as well, which would be distracting because VeraBank is not intending to fix most of them.]

When finding a vulnerability with Veracode Greenlight, the developer can read a short description of how to fix the vulnerability, including 'recommended fix' code snippets. If that is not enough, she can view a short **Veracode eLearning Tutorial** specific to the language and vulnerability. If there's still any lack of clarity, she can schedule a consultation call through **Veracode Application Security Consulting**.

Veracode Greenlight is also tied into the continuous integration pipeline and scans only the source code files that have changed before each deployment, failing the build if medium or higher vulnerabilities are detected. This helps the team to keep up their velocity to beat the competition.

The team is comfortable deploying small code changes with lighter but faster security scanning during the day. However, each night, all microservices are scanned with a **full policy scan** using **Veracode Static Analysis**, which ensures that the entire application is scanned as a whole.[3] This may uncover more complex vulnerabilities, for example when source and sink are not in the same project file. These vulnerabilities are then flagged to developers as work items through their Azure DevOps boards and are addressed the next morning. This risk was deemed acceptable for this type of application by the security team.



Use case 1 – Microservices: Microservices are getting broken out of the monolith. Diagram shows process for microservices, which are written from scratch. Developers remove all medium and higher severity vulnerabilities as they code. Veracode Greenlight added as a pipeline scan breaks the build if medium or higher vulnerabilities are found. Veracode Static Analysis policy scans occur every night to ensure full application is covered.

[3] If VeraBank was not comfortable with doing a policy scan at this point, they could use a sandbox scan and promote it to a policy scan once the application has passed. This depends on the preferred workflow of the company and the downstream impact of failing policy.

Software Composition Analysis	Especially in its new microservices deployments, VeraBank uses a lot of open source .NET components that it downloads from **NuGet**, a .NET package manager. To scan the open source components of their code, they use a **PowerShell Script** to integrate **Veracode Software Composition Analysis** into **Microsoft Azure DevOps**. The PowerShell script downloads the current version of the **Veracode Software Composition Analysis agent** and runs it on the code, reporting results back into the Veracode Platform.
Dynamic Analysis	To catch any configuration errors and other runtime security issues on the VeraCoin platform, VeraBank runs nightly dynamic scans using **Veracode Dynamic Analysis**. They run these scans only on production to check for configuration vulnerabilities. Note that it wouldn't make sense to scan both in pre-production and production because this team scans asynchronously, meaning they release while the scan is still running and catch security issues later.
Penetration Testing	VeraBank conducts a penetration test on the platform after each major architecture change, but at minimum once a year. Because they have opted for **Veracode Manual Penetration Testing**, their test results for this application appear on the same Veracode Platform portal as the automated assessments and are measured against the same policy they set up for the automated assessments. This makes it easy for them to see if an application passed policy because they don't have to manually compare results against their policy. Penetration testing results also automatically get promoted to Splunk. Having all assessment results in one place dramatically simplifies their audits.

USE CASE 2: RETAIL BANKING SOFTWARE IN JAVA STACK WITH ON-PREMISE DEPLOYMENT

VeraBank's retail banking software is a monolith web app developed in **Java**. They release updates to the software every two weeks. Their software has significant technical debt with many security vulnerabilities. Due to the criticality of the application, and because it won't be replaced or re-architected any time soon, they have decided to burn down their technical debt, but they also want to ensure that they don't introduce new vulnerabilities once they have remediated all medium and higher flaws.

The development team uses **Eclipse** as an IDE. They use **Jenkins** as a build server to deploy to an on-premise **Apache Tomcat** server. Bugs, including security issues, are tracked in **Atlassian JIRA** and assigned to the relevant developers. Their code is stored in a **Git** repository.

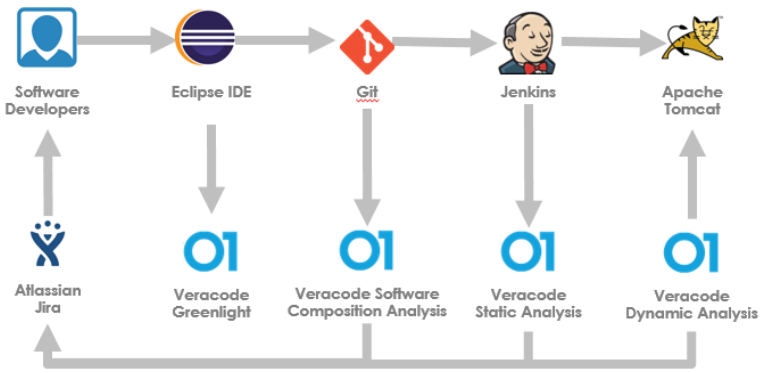
Static Analysis	VeraBank uses **Veracode Static Analysis** to scan the application as one big application profile. They scan feature branches first in the **developer sandbox** to enable the development teams to see results without promoting them to Splunk. The developers do this directly from the Veracode **Eclipse extension**. When the sandbox scans succeed, they merge the feature branch to the main branch. The release manager then runs a policy scan before releasing the software. This process is fully automated through Veracode's integration with **Jenkins**. After they had fixed all of the medium and higher flaws, they mitigated all of the remaining existing flaws as "acceptable risk" so that lower priority tech debt would not fail policy. The developers were effectively able to prioritize which vulnerabilities to fix first by looking at the **Fix First Analyzer**, which classifies all vulnerabilities by risk level and effort to fix.

Static Analysis Once Tech Debt Has Been Addressed

Over the first three months, the team has dedicated 20% of their capacity to reduce the technical debt so that the application is now passing policy. After celebrating this achievement, they decided that they would like to put measures in place to help them address security issues before they are released. Fixing vulnerabilities earlier in the SDLC significantly reduces cost and unplanned work, so they have added the **Veracode Greenlight** extension to all Eclipse IDEs, which notifies them about vulnerabilities in their code before they even commit it to the Git repository.

Several things are important to note: The team waited to roll out Veracode Greenlight until they had burned down their technical debt because they would otherwise have had to deal with too much noise in their IDEs. Going forward, they are only paying attention to vulnerabilities that are showing up in newly written parts of the code, so Veracode Greenlight is a good fit. Also, they are not using the Veracode Greenlight API in their pipeline because they are not releasing several times a day, so the sandbox scans are sufficient in providing automated results.

The regular scans using **Veracode Static Analysis** still continue to ensure a full application audit before each feature branch pull request and each release.



Use Case 2: Java stack application where developers first down technical debt with Veracode Static Analysis and then use Veracode Greenlight to ensure that they don't add new medium and higher severities to the code. Open source code is scanned with Veracode Software Composition Analysis. Veracode Dynamic Analysis is run on both production and pre-production.

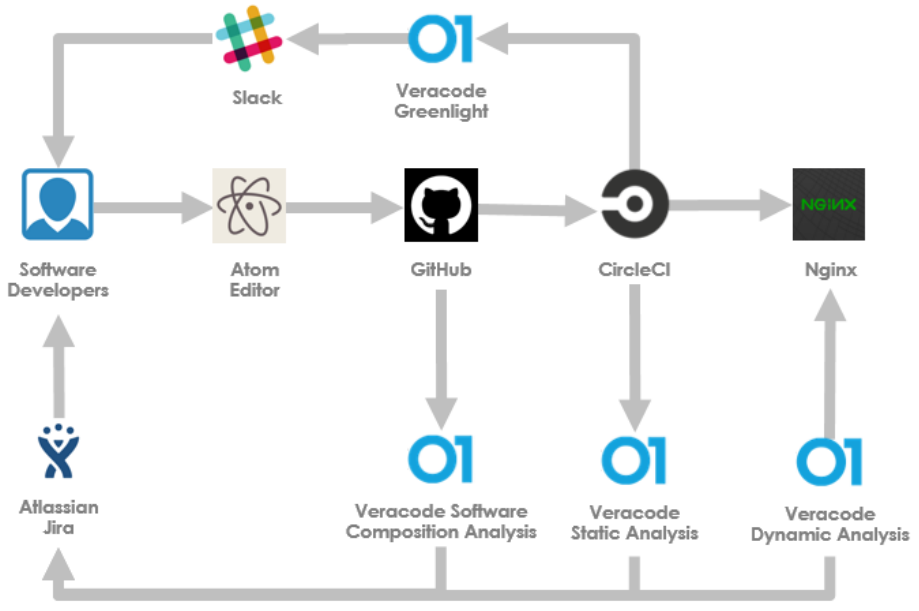
Software Composition Analysis	The retail banking software is heavily leveraging Java open source components, which have helped the team move faster because they don't have to reinvent the wheel on standard functionality. To scan the open source code for vulnerabilities with **Veracode Software Composition Analysis**, the team chose to use the **binary upload method** rather than the SCA agent. This saves them one step because the binaries are already uploaded to the Veracode Platform.
Dynamic Analysis	The release engineer runs dynamic analysis scans using **Veracode Dynamic Analysis** on the complete application in pre-production before it is released. However, she also scans the application in production to catch any drift between configurations in pre-production and production.
Penetration Testing	This application changes relatively little over the course of a year, so VeraBank opts for a single penetration test per year through **Veracode Manual Penetration Testing**.

USE CASE 3: VIRTUAL ASSISTANT MICROSERVICE USING JAVASCRIPT

VeraBank's website front-end is written in **JavaScript**. The project's developers use the **Atom editor**. The pipeline is built on **CircleCI**. They use **JIRA** as a ticketing system but most communication on the team happens via **Slack**. **GitHub** is the code repository. Releases happen once a day, typically in the morning so that the team can handle any production issues while they are all in the office.

Static Analysis in the Pipeline	This team has been building in application security from the start. The wanted to scan as early as possible in the process. Because they are using the **Atom editor**, for which there is no plug-in for Veracode Greenlight, they are building the **Veracode Greenlight API** directly into **CircleCI.** Any flaws that are found after a check-in are automatically posted to the *#pipeline* **Slack** channel through a custom script written by the DevOps engineers; the build breaks for any flaws of medium or higher.
Static Analysis for Auditing	The team is comfortable deploying small code changes with lighter but faster security scanning during the day. However, each night the microservice is scanned with a full policy scan on the code running in production using **Veracode Static Analysis**, which ensures that the entire application is scanned as a whole. This may uncover more complex vulnerabilities, for example when source and sink are not in the same project file. These vulnerabilities are then flagged to developers in a JIRA ticket and are addressed the next morning. This risk was deemed acceptable by the security team.
Static Analysis Once Tech Debt Has Been Addressed	The team has worked to reduce the technical debt so that the application is now passing policy. They decided that they would like to put measures in place to help them address security issues before they are released. Fixing vulnerabilities earlier in the SDLC significantly reduces cost, so they have added the **Veracode Greenlight** extension to all Eclipse IDEs, which notifies them about vulnerabilities in their code before they even commit it to the Git repository. Like in the previous use case, the team waited to roll out Veracode Greenlight until they had burned down their technical debt because they would otherwise have had to deal with too much noise in their IDEs. Going forward, they are only paying attention to

	vulnerabilities that are showing up in newly written parts of the code, so Veracode Greenlight is a good fit. Also, they are not using the Veracode Greenlight API in their pipeline because they are not releasing several times a day, so the sandbox scans are sufficient in providing automated results.
Software Composition Analysis	The microservice uses a lot of JavaScript open source components, which they are scanning with **Veracode Software Composition Analysis** through an integration with **CircleCI** with every commit. Security issues are dispatched to the developers via **JIRA** based on priority thresholds defined in the policy.
Dynamic Analysis	VeraBank runs nightly pre-production dynamic scans using **Veracode Dynamic Analysis.** The scans complete overnight so that results are ready before deployment in the morning.
Penetration Testing	VeraBank has pentested this API through **Veracode Manual Penetration Testing** before it was first released; the next annual test is already scheduled.
Security Attestation	VeraBank's customers are very security conscious, especially since VeraBank's competitor had a data breach last year. VeraBank uses this as a marketing opportunity and has listed the online banking app in the **Veracode Verified Directory**, showing third parties that the application has undergone security scrutiny. It has also used the **Veracode Verified Seal** on its website to promote the higher security level of its app. VeraBank loved that this attestation was included in their licenses for their existing Veracode solutions.



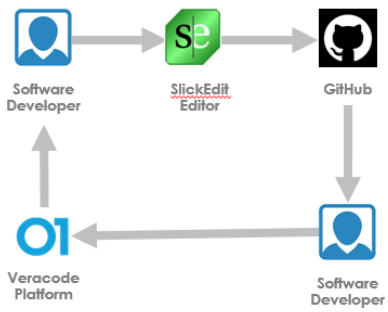
Use Case 3: Veracode Greenlight scans code from CircleCI with every pipeline build and is integrated with Slack through a custom script to deliver results right back to the developers. Results from Veracode Software Composition Analysis, Veracode Static

Analysis, and Veracode Dynamic Analysis are fed back to JIRA and reported to developers.

USE CASE 4: LEGACY COBOL APPLICATION ON MAINFRAME

VeraBank started out in the 1970s, so it also has some legacy applications, such as its **Mainframe** application for trading pork belly futures. The application is written in **COBOL** and only updated twice a year, so it does not make sense for VeraBank to keep a full-time COBOL programmer on staff. Their development is outsourced to Ansgar, a freelancer who just uses **SlickEdit** as an editor. He's uploading the COBOL code to a private VeraBank **GitHub** account.

Static Analysis	When Ansgar has updated and tested his COBOL app, he downloads the full source code from **GitHub** and uploads it to the **Veracode Platform** to kick off a **sandbox scan** in **Veracode Static Analysis.** He views all flaws directly in the Veracode Platform. If the application passes policy, he promotes the sandbox scan to a policy scan and lets VeraBank know that they can deploy it into production. Although these processes are manual, it's not worth it for Ansgar to automate them because he only has to go through the process twice a year. If he wanted to automate them, he could use Veracode's API wrappers to automate the upload and get flaws delivered as tickets in JIRA.
Penetration Testing	VeraBank has scheduled a pentest through **Veracode Manual Penetration Testing** once a year to ensure that the mainframe application is secure. The Veracode penetration tester uses the results from the static analysis scan to go deeper, faster, providing a higher-quality penetration test than a simple black box test.



Use Case 4: In this no-frills scenario, the developer uploads the COBOL code manually to the platform and views the scan results right there.

VERACODE COVERS YOU FROM LEGACY APPLICATION TO BLEEDING EDGE

We hope that this document painted a picture of the flexibility of the Veracode Platform. Just as every jazz song is interpreted slightly differently every time, every customer scenario is slightly different. After 11 years in the business, Veracode has seen a lot of different scenarios in a constantly evolving landscape, so we have crafted a solid platform with many assessment types and even more integrations.

How exactly you deploy Veracode depends on your programming languages, the types of applications, your development cadence, your development tool chain, your security stack, and each application's level of acceptable risk.

VERACODE

If you are still shopping for an application security vendor, please discuss with your Veracode Solutions Architect what she recommends for your various teams. If you are already committed to us, please work with your Veracode Security Program Manager to develop a plan on how to onboard each one of your teams. We thank you for your business!