Checkmarx

WHITEPAPER

# Building Secure Applications: Recommendations for Financial Services

# Introduction

# Regulatory Requirements for Financial Services

Financial services organizations operate under a host of regulatory standards. This makes sense, as the assets and information managed by these firms are valuable, sensitive, and targeted by sophisticated cyber attackers daily. Compounding these challenges is the large volume of personally identifiable information (PII) that financial organizations handle regularly.

PII is subject to many regulations and standards, particularly Graham, Leach, Bliley (GLBA), the Payment Card Industry Data Security Standard (PCI-DSS), and the Sarbanes Oxley Act (SOX). Today, the General Data Protection Regulation (GDPR) is also top-of-mind, as it regulates not only the processing of personal data, including PII of individuals in the European Union, but also for any organization that processes personal data of EU residents. For U.S. banking consumers, Section 5 (Unfair or Deceptive Acts or Practices) of the Federal Trade Commission Act and numerous state regulations enforce basic consumer protections, which financial organizations must also uphold.

The regulatory landscape for financial services organizations continues to expand. The intent, of course, is to protect customers' investments and sensitive information as well as preventing fraud. Organizations subject to these regulations and standards must navigate a wide range of stringent requirements.

Some are quite explicit; the PCI-DSS includes specific activities organizations must follow (for example, the use of manual or automated processes to identify common vulnerabilities such as SQL injection, cross-site request forgery, cross-site scripting, buffer overflows, and others). Other regulations are quite general, simply stating that PII must be secured from attacks.

While the various regulations and standards take different approaches, they require the same reasonable security approaches. Most notably, to comply with any of the major regulatory standards, organizations must have visibility into risks and vulnerabilities in their software and systems (by conducting regular vulnerability assessments) and a plan for addressing the vulnerabilities (by establishing and following a vulnerability management plan).

Vulnerability assessments can occur in a number of ways. For example, there are commercial solutions for scanning environments to identify unpatched or misconfigured applications or systems. While useful, these solutions focus on commercial software and operating systems such as Linux and Windows. These types of scans are blind to the thousands of in-house applications built and maintained by most financial services organizations. In-house applications require a different set of skills and solutions for identifying and addressing vulnerabilities.

# The Changing Application and Software Development World

Recently, software development has dramatically changed in three significant ways, forcing application security to change as well.

### The Shift from Waterfall Software Development Methodologies

In the past, most software development projects used a waterfall development model. In this model, teams followed a structured process of requirements gathering, software design, implementation/coding specific to a release, verification/testing of that release, and ongoing maintenance. The development life cycle might last for months, with two or three major releases each year. Application security testing, if done at all, was conducted by autonomous security teams late in the development life cycle.

While still used for some larger, legacy applications, organizations now view quicker software development cycles as a way of gaining a competitive advantage, and increasingly, as a business requirement.

These faster software development methodologies, such as Agile, focus on integrated teams including software architects, developers, and functional and security testing teams working together to deliver usable features as quickly as possible.

By listening closely to customers – and implementing newly discovered requirements quickly – organizations may gain market share against slower competitors.

### The Dawn of DevOps Deployments

Using Agile development processes can help organizations build and deliver software faster. Adding DevOps concepts such as Continuous Integration and Continuous Delivery (CI/CD) into an Agile environment helps break down silos by integrating software development and software operations, increase quality and efficiency, and make incremental changes available to users more quickly. Continuous integration (CI) refers to the process by which new code is made part of the mainline codebase with minimal delays. Continuous delivery (CD) is a software development practice in which every code change goes through the entire pipeline and is made available to end users quickly.

The keys to a successful DevOps migration requires the use of appropriate tools and automation. While the tools may differ by organization, a continuous integration (CI) pipeline such as Jenkins or TeamCity is critical to successfully automating processes. For example, a CI pipeline allows organizations to automate security testing during the build process and also leverage existing functional testing with security automation during the CD process.

## The Rise of Open Source

Years ago, in-house software development teams created custom software from top to bottom. Open source components were viewed as dangerous and were rarely used due to their unknown origin and unusual licensing models. As the open source community grew, so did acceptance of popular open source projects including Linux, OpenSSL, and frameworks such as Apache Struts. Today, open source software often comprises the majority of codebases in both in-house and commercial software.

Open source components and frameworks offer many benefits, such as eliminating the need to write common functions from scratch, thereby lowering development costs and accelerating time-to-market. But open source deployments also introduce new risks to an organization. Each year, more than 3,000 new vulnerabilities are disclosed in open source. Often, exploits for these vulnerabilities are publicly available within days of public disclosure – providing cyber attackers with a simple attack vector without much effort. In 2017, this risk was dramatically demonstrated in a high-profile breach, when a publicly disclosed vulnerability in Apache Struts was exploited. The hackers stole PII of more than 148 million consumers, resulting in a $6 billion loss for the organization's market capitalization, and the dismissal of the company's CEO, CIO, and CSO.

**10**

### Benefits of DevOps for Financial Services Organizations

1. Reduces time-to-market
2. Automates manual, labor-intensive processes
3. Fosters cross-team collaboration
4. Increases operational efficiency
5. Improves overall performance
6. Streamlines compliance and simplifies audits
7. Reduces development and IT infrastructure costs
8. Improves quality
9. Enhances the customer experience
10. Reduce downtime, failures, and rollbacks

# Security Testing Must Adapt to the New Paradigm

This new software development paradigm requires security teams to adapt, as delays caused by "out of band" – or new/different – processes will effectively break the DevOps model. This starts with clearly defining and building bug identification and threat remediation directly into everyday processes – not bolting them on at the end. Functional and security testing must also occur automatically as part of the process. The speed of DevOps cannot tolerate separate testing cycles. And it's important to remember that when it comes to DevOps, security expertise and tooling are just as critical as CI orchestration or functional testing.

Full visibility into the open source components in internally developed software is also required to effectively secure the modern software development life cycle (SDLC). Application Security Testing (AST) solutions, including static and interactive analysis, are exceptional at identifying coding errors that may result in security vulnerabilities in custom code. Unfortunately, static and interactive analysis solutions are ineffective at identifying even previously disclosed vulnerabilities in open source software. To address the open source portion of the codebase, software composition and open source analysis solutions are required. These solutions scan software to produce a list of all open source components in use, then map those components to vulnerability databases to identify components with known vulnerabilities.

# Five Steps to Building More-Secure Applications

While financial service organizations are under constant attack from adversaries, there are specific steps they can take to address security in the software they create.

### 1. Risk Rank Applications

Many financial services organizations have software development teams that dwarf those of large commercial software companies. In fact, some global banking organizations have over 20,000 software engineers building and supporting thousands of individual applications. Securing the modern software development life cycle is an extremely challenging role for today's financial organizations. With an ever-expanding cyber attack surface, security teams often feel like they need to "boil the ocean" so to speak. But from a business risk perspective, not all applications are equal.

The first step in reducing risk is to quantify the inherent risk associated with each application. This can be accomplished by using a risk-prioritized methodology to rank applications based on potential damage to the firm's business goals as a result of a successful attack. For example, the security of an online banking application that allows customers to transfer funds, perform large transactions, and change privileges is crucial to a bank's business goals. A breach of that application could cause financial, regulatory, and reputational damage to the bank. Likewise, applications that manage information subject to security standards such as PCI-DSS are viewed as critical and must be secured.

However, there are internal applications that do not process sensitive information or have a limited attack surface. In terms of business value, these applications are less critical and do not warrant the same scrutiny from a security standpoint.

Risk ranking applications can empower time- and resource-constrained security teams within financial services organizations to apply appropriate resources to the applications with the most risk, while maximizing operational efficiency.

## 2. Establish Clear Security Requirements

To achieve true "DevSecOps," developers, security teams, and operations teams must agree in advance on the metrics for adequate security. This requires open and ongoing communication and collaboration between teams. Metrics will differ for various application types, based on risk ranking and the organization's unique appetite for risk.

For open source components, these requirements must include an understanding of each project, including:

- How well a project is supported by the community. An unsupported open source project can result in unplanned management and ongoing maintenance for the internal IT operations team.
- The component's security history. It's important to review how many vulnerabilities have been disclosed to date and how quickly the community fixes them.
- The open source licenses requirements. It's equally important to identify associated software licenses to uphold compliance and adhere to restrictions.

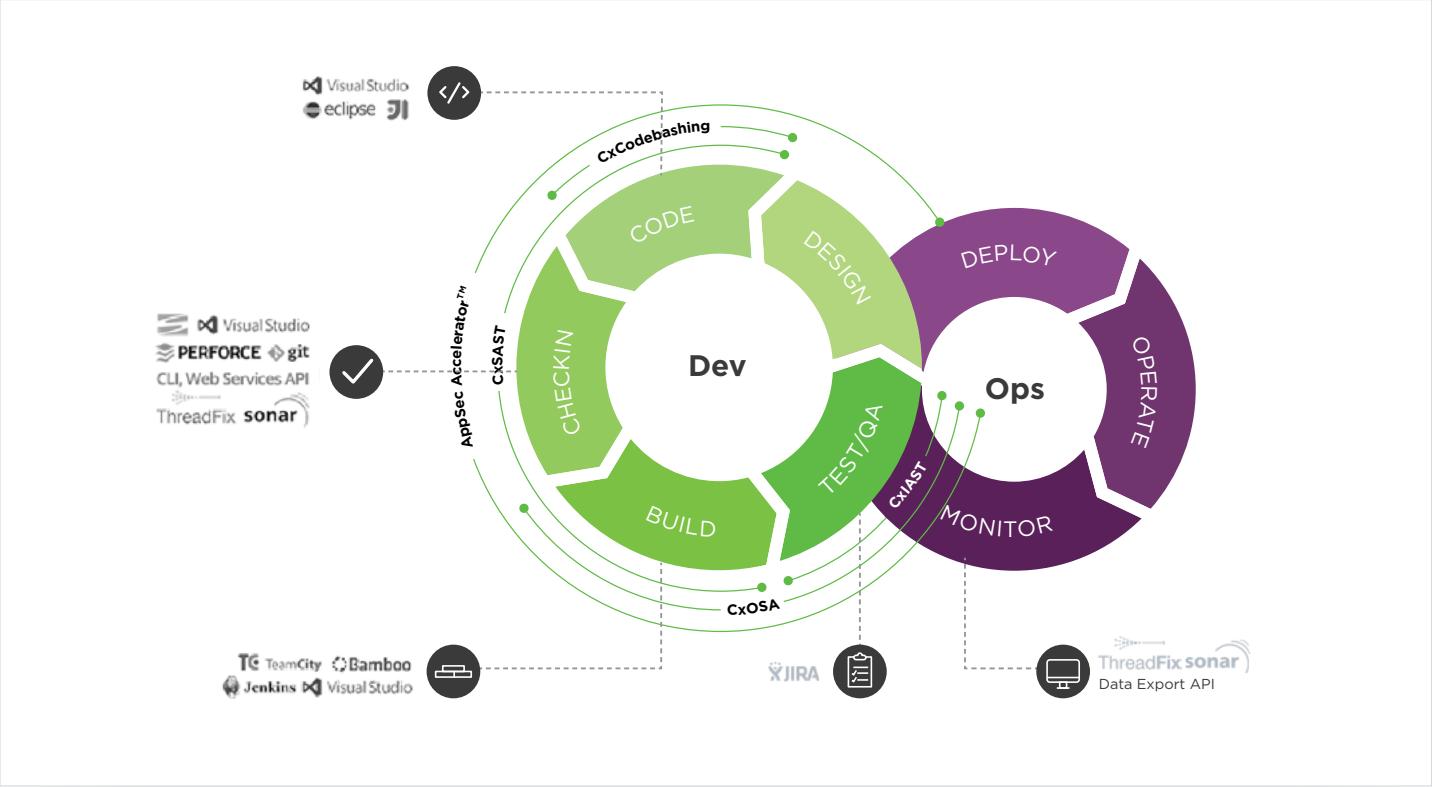
For custom code and the complete application, it's essential to have an agreement in place explicitly stating when security testing will occur and what conditions will necessitate breaking a build. For example, an organization may dictate that applications cannot be deployed if a "severe" vulnerability is identified. Some may elect to stop a build when that condition is found, while others may choose to allow a build to continue even with severe vulnerabilities, if it is not slated for release to a production environment.

## 3. Identify Vulnerabilities Throughout the SDLC

Security must be integrated into all phases of the software development life cycle. This approach will not only improve security in DevOps environments, but will also accelerate time-to-market and lower development cost, since vulnerabilities found earlier in the SDLC are usually less complicated and less costly to remediate.

Static application security testing (SAST) solutions integrate into the SDLC from the beginning of the code phase, through check-in and build. Open source analysis (OSA) can be used in the earliest builds to identify open source dependencies and map those components to publicly disclosed vulnerabilities, continuing through the test/QA phase. Integrated application security testing (IAST) solutions used during functional testing in the test/QA stage are also required. It's critical to have SAST, OSA, and IAST solutions that integrate into the CI orchestration so teams can automate processes and perform incremental scans of only the code that has changed. Solutions that require hours to scan a complete build do not fit well into a DevOps environment.

For in-house software, security teams must remember that open source and custom code require different testing methodologies to identify risk and gain true visibility into the security of their software.




The diagram above highlights where the Checkmarx solutions fit within DevOps.

- **CxSAST -** Static Application Security Testing
- **CxOSA -** Open Source Analysis (also know as Software Composition Analysis)
- **CxIAST -** Interactive Application Security Testing
- **CxCodebashing -** Gamified Secure Coding Education
- **AppSec Accelerator -** Managed Software Security Testing

### 4. Empower Developers to Code Securely from the Start

To address security at the beginning of the development life cycle, it's important for security teams to take an active role in engaging and collaborating with their DevOps counterparts. Education is a huge part of this. Security teams should train DevOps teams on specific attack methods and popular hacking techniques, provide the educational tools they need to identify vulnerabilities as they write code, and act as a sounding board throughout the process. By providing ongoing feedback and being available to answer secure coding questions on demand, security teams can greatly reduce the time required to fix vulnerabilities, resulting in better security and more predictable software delivery. By establishing best practices and making Secure Coding Education (SCE) an ongoing process, security teams can make it easy for developers to code securely from the start. Further, developers will be more receptive to training when it's relevant, retain lessons learned, and ultimately, become security champions for the organizations.

### 5. Remember that Application Security Is Not a Once-and-Done

Open source components and frameworks offer clear advantages, including lowering development costs and accelerating time-to-market. To maintain strong security, open source components must be analyzed during the coding and building phases. But it can't end there. It's critical to continue monitoring open source software for newly disclosed vulnerabilities throughout the SDLC. Some vulnerabilities such as ShellShock (CVE-2014-6271) were discovered decades after the original vulnerability was created. Without visibility into both the version of the open source component and its location in the codebase, it's impossible to find and fix those vulnerabilities. Effective application security should be continuous.

# Conclusion

Today, malevolent actors deliver PII used for identity theft in vast quantities, while impacts of a data breach go far beyond embarrassment. Today's breaches cause loss of reputation, significant loss to shareholder value, and even dismissal of corporate leadership. These breaches also bring significant fines due to ever-increasing regulations, along with heightened legislative inquiries, and public distrust.

The way financial services organizations build software today is dramatically different than just 10 years ago. New development models deliver software faster than ever before to meet changing consumer demands, maximize operational efficiency, and drive digital transformation. In the highly competitive financial services market, it is simply no longer an option to deliver software that hasn't been tested for security issues throughout the development process. The risks are far too great. Software is everywhere, and users rely on both the software itself and its security to complete billions of transactions a day. It's time to build security in from the start of the SDLC to better manage, measure, and address risk, empower development teams, and guarantee secure software delivery at the speed of DevOps.


## About Checkmarx

Software Security for DevOps and Beyond.

Checkmarx makes software security essential infrastructure: unified with DevOps, and seamlessly embedded into your entire CI/CD pipeline, from uncompiled code to runtime testing. Our holistic platform sets the new standard for instilling security into modern development. Learn more **here**.