EBOOK

Contrast SECURITY

How to Secure APIs at DevOps Speed

Executive Overview

As development operations (DevOps) increases the speed at which applications go to market, many new vulnerabilities are finding their way into production. Application programming interface (API) vulnerabilities have become an exceptionally popular target for attackers, due to the limitations of traditional security solutions. In response, an instrumentation-based application security (AppSec) approach can help automate security responses inside the application itself for more effective protection and more efficient operations.

This eBook shows how an instrumentation-based approach to security—protection that is built into the application itself—can help automatically detect and protect API vulnerabilities, simplify deployment, scale to accommodate growing DevOps volumes, and enhance ongoing management processes.

> According to Gartner Research, 90% of web-enabled applications will be more exposed to attack by API weaknesses than via the user interface—up from 40% of apps in 2019. Indeed, within two years, Gartner predicts that APIs will be the most commonly targeted attack vector in the enterprise.[1]

[1] Dionisio Zumerle, et al., "How to Build an Effective API Security Strategy," Gartner, December 8, 2017.

Table of contents

01

DevOps Delivers Speed – But Still Needs Security

According to Verizon research, applications have been the most common vector for hacking-based breaches in recent years.[2] As a result of these widespread vulnerabilities, the AppSec market is expected to reach $7.1 billion in annual spend by 2023 (a compound annual growth rate of 16.4%).[3]

While a great many organizations have embraced DevOps to accelerate the development and delivery of new applications, the need for speed has come at the cost of security. Application vulnerabilities continue to be exploited by an increasingly sophisticated array of exploits and threat actors. Hackers can steal data faster than ever before. And businesses are often left to pay a very high price for losing control of sensitive data and private records as a result of weak points within the application stack—whether in the code itself, an open-source library, or (very often) in an API.

> 67% of global security technology decision-makers rated improving application security as a high or critical priority—the highest of all tactical priorities.[4]

[2] "2019 Data Breach Investigations Report," Verizon, April 2019.

[3] Amy DeMartine, et al., "Application Security Market Will Exceed $7 Billion By 2023," Forrester, updated March 19, 2019.

[4] Amy DeMartine, et al., "Application Security Market Will Exceed $7 Billion By 2023," Forrester, updated March 19, 2019.

APIs, in particular, present an increasingly attractive target to attackers. APIs are used to connect applications and extend functionality. They are the formal, regularly stated ways that pieces of applications talk to one another. This means there is at least one API for every component in an application.[6] As a result, APIs are everywhere, and they are growing in numbers—the average organization manages over 300 APIs, many of which are exposed externally to customers and partners.[7]

But at the same time, increasing demands for speed of delivery and lack of time and/or resources due to workloads are the two biggest obstacles for ensuring API quality.[8] API configuration errors or coding vulnerabilities can be exploited to exfiltrate valuable data such as the personally identifiable information (PII) of customers or company intellectual property (IP). As Gartner notes, while 70% of enterprises consider APIs to be important to digital transformation, they also admit that security remains a key challenge.[9] In recent years, private and public sector organizations have all fallen victim to attacks on vulnerable APIs.[10]

> *Hundreds, if not thousands, of potential vulnerabilities exist in the places where applications and functions come together—namely, APIs.[5]*

> *With the right attack, a wealth of valuable data can be exfiltrated via APIs— ranging from customer personally identifiable information (PII) to company intellectual property (IP).[11]*

[5] Curtis Franklin Jr., "How to Manage API Security," Dark Reading, December 17, 2019.
[6] Curtis Franklin Jr., "How to Manage API Security," Dark Reading, December 17, 2019.
[7] Roey Eliyahu, "5 Things You Need to Know About API Protection," SC Magazine, April 16, 2019.
[8] "The State of API 2019," SmartBear, February 2019.
[9] Dale Gardner, "Market Insight: Product Strategy Considerations for the Emerging API Threat Protection Market," Gartner, August 2, 2019.
[10] Roey Eliyahu, "5 Things You Need to Know About API Protection," SC Magazine, April 16, 2019.
[11] Roey Eliyahu, "5 Things You Need to Know About API Protection," SC Magazine, April 16, 2019.

Contrast SECURITY

02

API Vulnerability Requires Enhanced Visibility

API vulnerabilities are like any other AppSec problem. The cost of fixing defects increases exponentially later in the software development life cycle (SDLC). It costs six times more to fix a bug found during implementation than to fix one identified during design; 15 times more if it is identified in testing; and 100 times more during regular maintenance once the code is in production.[12] Therefore, organizations need to "shift left" and strive to fix as many problems as they can as early as possible in the SDLC.

Unfortunately, traditional AppSec testing methods in preproduction have significant limitations—especially when it comes to APIs. Static application security testing (SAST) and dynamic application security testing (DAST) tools are inherently slow, and their approach to checking every line of code is often redundant. Worst of all, they are not designed to secure APIs.[14]

But an instrumentation-based AppSec solution with route intelligence capabilities can solve these problems. Rather than testing and re-testing lines of code, route intelligence maps URLs to code paths to inform security testers of how the application is accessed, as well as if they have actually tested each route. It then maps vulnerabilities to routes, enabling teams to understand the level of authorization per route and accessibility of any discovered vulnerabilities.

In this way, a solution with route intelligence can expose all of the different points of entry into the application—including APIs. As a result, no additional specialized testing needs to happen to capture vulnerability assessments. This ensures advanced discovery, visibility, and verification.

Security remains a major concern for 43% of API developers, testers, and product leaders.[15]

> *"Developers have taken to APIs as a way to connect applications, extend functionality, and interface with partners. This creates an often-complex web of logic, connectivity, and exposure for critical infrastructure and data. It also generates new vulnerabilities and new targets for attackers.[13]*

[12] Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 16, 2020.
[13] Roey Eliyahu, "5 Things You Need to Know About API Protection," SC Magazine, April 16, 2019.
[14] Jeff Williams, "What Do You Mean My Security Tools Don't Work on APIs?!!," Dark Reading, June 25, 2015.
[15] "The State of API 2019," SmartBear, February 2019.

Contrast SECURITY

03

Automation Helps Unify DEV + SEC + OPS

In addition to shifting left to help fix critical API vulnerabilities early in the SDLC, organizations must also account for the fact that new exposures often appear post-release. Therefore, security teams must also "shift right" to automatically protect application vulnerabilities, including those involving APIs, that appear in production.

Embedding an instrumentation-based AppSec agent within the application code itself facilitates better collaboration between development and security teams (DevSecOps). The agent is deployed with the code at the beginning of the development process and only sends notifications as problems arise. When on the inside, security sensors have unobstructed API visibility—leveraging internal knowledge about what data structures the API accepts.

This enables better accuracy and performance because the sensors do not analyze data in transit that will simply be discarded or rejected by the application. All parts of the application's actual runtime usage (including APIs) are visible and protected by the AppSec solution.

Because most usage is safe, the security agent does not interfere or delay the development process. This allows developers to move at their own pace, without any potentially adversarial back-and-forth checks with a security team. Instead, embedded security moves with the application through all stages of its life cycle—from development to deployment and beyond.

> *Securing DevOps is about more than technology. Marrying security and DevOps requires a change in culture, a breaking down of barriers, and dispelling the myths of misguided beliefs that have led to finger-pointing in the past.*[16]

> *More automation means less risk of security vulnerabilities caused by human error. And if something does go awry, automation can make the problem easier to pinpoint and fix.*[17]

[16] Kacy Zurkus, "Securing DevOps Is About People and Culture," Dark Reading, August 6, 2019.
[17] "How to Leverage DevOps and Automation to Bolster Security," Tripwire, July 7, 2019.

04

Outside Security vs. Inside Security

As anyone (a legitimate user or a potential attacker) goes to make a request into an application, they make that request through the **open (public) API**. The request then passes through **(WAFs)** to check for any previously known threats. It then goes through an application gateway that figures out where the request is supposed to go. It may pass through a load balancer before making its way to the server or serverless area before it hits the code written for the actual **implemented (private) API**—which triggers other resources such as databases and files inside the application.

Many organizations currently depend solely on "outside" perimeter-based security gates—such as WAFs—to protect deployed applications. While these solutions can defend against known threats, they are not capable of catching unknown or zero-day threats. These sorts of **false negatives** allow threats inside the application where often there are no other defense mechanisms in place. This, in turn, can lead to a successful attack that disrupts or damages the business and requires substantial remediation effort.

Traditional security at the perimeter also has accuracy problems that generate a lot of unnecessary manual analysis and remediation. Here, WAFs generate noise from **false positives**—probes that blindly check for vulnerabilities that may or may not exist in a given application. In order to sort actual threats from the false alarms, security staff needs to manually investigate the alert. Because of these manual process requirements, perimeter security solutions (i.e., WAFs) cannot support automated, on-demand self-service for AppSec.

In addition, if the application, including APIs, changes or needs to be redeployed for any reason, perimeter security solutions need to be re-tuned or run a new learning period for that deployment. Further, because these critical processes rely on people, this also opens the door for human errors that introduce risks and degrade security effectiveness.[18]

An instrumentation-based approach for "inside-out" security not only supports automation but it offers protection in a runtime context—visibility across all parts of the running application. If APIs, libraries, or containers are vulnerable, then the application inherits those vulnerabilities. And even if a component is not vulnerable today, it may become exposed tomorrow. As a result, vulnerabilities should be expected.

With security on the inside, sensors monitor for vulnerabilities in applications and APIs as they appear and accurately block only the attacks that attempt to exploit them. If something is not an actual threat, it can be ignored.

An effective instrumentation-based AppSec solution should include:

- **Software composition analysis (SCA),** which tracks app components (including APIs) for known issues such as open-source vulnerabilities
- **Interactive application security testing (IAST),** which automatically watches running code for security vulnerabilities—without requiring the involvement of a security expert
- **Runtime application self-protection (RASP),** which improves security performance in production by only blocking the attacks against actual vulnerabilities of the specific application—including exploits that target APIs

> *By deploying comprehensive DevSecOps automation, organizations gain visibility and control over the development life cycle, along with a closed-loop pipeline for testing, reporting, and solving for potential security concerns.*[19]

[18] "Perimeter Security Noise Leaves Applications Vulnerable to Attacks," Contrast Security, April 2020.

[19] "How to Leverage DevOps and Automation to Bolster Security," Tripwire, July 7, 2019.

05

Keeping APIs Safe— Right Where They Live

As it is the case for DevOps, automation is the path toward actualizing an effective DevSecOps environment. Embedded security instrumentation offers a path toward automating security tasks and unifying development teams. It provides more accurate and effective security across all parts of the application runtime—including frequently targeted APIs. And at the same time, this approach also helps staff dedicate their resources toward developing new products and solving more critical problems.

> *API abuse is an ongoing problem and is expected to escalate in the coming years, as the number of API implementations continues to grow.*[20]

[20] Zeljka Zorz, "Security Pitfalls to Avoid When Programming Using an API," Help Net Security, January 14, 2020.

Contrast Security provides the industry's most modern and comprehensive Application Security Platform, removing security roadblocks inefficiencies and empowering enterprises to write and release secure application code faster. Embedding code analysis and attack prevention directly into software with instrumentation, the Contrast platform automatically detects vulnerabilities while developers write code, eliminates false positives, and provides context-specific how-to-fix guidance for easy and fast vulnerability remediation. Doing so enables application and development teams to collaborate more effectively and to innovate faster while accelerating digital transformation initiatives. This is why a growing number of the world's largest private and public sector organizations rely on Contrast to secure their applications in development and extend protection in production.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com