EBOOK

Contrast SECURITY

How Manual Application Vulnerability Management Delays Innovation and Increases Business Risk

Executive Overview

With 62% of data breaches and 39% of incidents occurring at the web application layer,[1] identifying and remediating these errors as quickly as possible is a primary concern for an organization's security team. However, development teams have other priorities—namely, driving digital transformation forward by ensuring that code commits and product releases are completed on schedule. Neither the security nor the development team should compromise on their key business objectives.

Traditional approaches to application security (AppSec), such as static application security testing (SAST) and dynamic application security testing (DAST), lack visibility across an application's attack surface. As they analyze lines of code using brute force or look for code vulnerabilities based on a predetermined malware signature list, SAST and DAST approaches miss false negatives while incurring high volumes of false positives. Further, with significant volumes of cyberattacks employing unknown—or zero-day—threats, SAST and DAST simply are unable to protect modern software. Visibility extends beyond challenges with vulnerability identification—namely, lacking visibility into software routes, developers must expend significant time searching for and verifying that vulnerabilities were fixed.

62% of data breaches and 39 percent of incidents occur at the web application layer.

[1] "2019 Data Breach Investigations Report," Verizon, May 2019.

Table of contents

01

When Competing Forces Collide:
Speed and Security

When Competing Forces Collide: Speed and Security

Development and security teams are frequently at odds. On one side, developers are focused on code commits, release dates, and timelines. Their performance metrics are based upon getting a product out the door—on schedule and on budget. In contrast, security teams are most concerned with preventing cybersecurity risks to the organization and ensuring that they maintain compliance with applicable regulations.

The majority of organizations struggle to bridge the gap between these viewpoints. As many as 52% of companies admit to cutting back on security measures in order to meet business deadlines.[2]

26%

of enterprise applications contain serious vulnerabilities.[3]

[2] "52% of Companies Sacrifice Cybersecurity for Speed—Webinar Recap," Threat Stack, March 13, 2018.
[3] "2020 Application Security Observability Report," Contrast Security, June 2020.

This conflict between the security and development teams creates significant security risks for an organization. If security is seen as an impediment to development, security testing may be performed in a cursory manner, if not skipped entirely, allowing vulnerable code to reach production.

This puts an organization, or any users of its products, at risk of cyberattacks that could result in operational disruption, data breaches, and brand damage. Additionally, a failure to identify and remediate vulnerabilities can incur penalties levied due to regulatory noncompliance.

> Integration of Security Testing into Development Processes Reduces Data Breach Costs by an Average of $10.55 Per Compromised Record.[4]

[4] "2019 Cost of a Data Breach Report," IBM Security and Ponemon Institute, July 23, 2019.

02

Vulnerability Identification with Brute Force using SAST and DAST

Vulnerability Identification with Brute Force using SAST and DAST

SAST and DAST are two very different approaches to application security testing. **SAST** takes a "white box" and signature-based approach to testing. These solutions attempt to build a model of an application and pseudo-execute it to guess the application's runtime behavior. However, it is unable to see how an application would actually perform at runtime.

DAST, on the other hand, is "black box" testing performed using an application programming interface (API). The application's API is subjected to a barrage of attempted attacks and malicious HTTP inputs in order to determine if an application has defenses in place to protect against them. Like SAST, these solutions lack visibility to the internals of an application.

A well-designed SAST and/or DAST solution is capable of identifying a variety of vulnerabilities within an application. However, they also have shortcomings that lead to both **false positives** and **false negatives:**

FOCUS ON LINES OF CODE CAUSES MISSED VULNERABILITIES

When searching for vulnerabilities, SAST solutions focus on lines of code, which creates false positives due to analysis of "**dead code**" within an application, which is never reached or executed.

Additionally, this approach can suffer from a high false-negative rate due to a lack of visibility into library code. As a result, they are unaware of the custom detection rules that should be applied to that library. For example, a library function may use a string passed as input when building a directory, creating a potential resource vulnerability. Without insight into the code, a SAST solution is unaware of the potential risk.

SAST solutions' treatment of bulk data structures, such as arrays, lists, and collections, also generates false positives since a single untrusted value in the array causes all of its outputs to be untrusted. Finally, SAST is heavily dependent upon the quality of its signature library and is incapable of detecting unknown and zero-day attacks.

TARGETING APIS OVERLOOKS CODE EXECUTION PATHS

DAST solutions perform analysis via an API. By testing an application against a library of known attacks, such as SQL injection and cross-site scripting (XSS), these analyzers can identify a range of vulnerabilities.

However, DAST solutions lack visibility into the internal workings of an application, meaning that they can overly test certain APIs and execution paths within the application and completely overlook the existence of others. The latter results in false negatives (vulnerabilities that are missed in analysis).

False negatives are not the only problem created by DAST solutions. These analyzers are also prone to false positives since they **alert on probes as well**, where a malicious input never reaches vulnerable code. Cyber criminals commonly use probes to detect potentially vulnerable applications; however, applications not vulnerable to the attack are unaffected. Forcing developers to fix errors delays code commits and software releases.

INACCURATE DETECTIONS WASTE DEVELOPER TIME

False positives and false negatives are very different, but they both result in wasted time for an organization. Each false positive that a SAST or DAST solution generates must be **manually investigated and remediated** by developers, delaying code commits and release cycles.

43%

of organizations cite API security as a concern.[5]

[5] "The State of API 2019," SmartBear, February 2019.

The impacts of false negatives, on the other hand, come later when a detected vulnerability in production code forces costly patch development, and potentially, incident response activities addressing breaches caused by exploitation of these vulnerabilities. Fixing a vulnerability in development costs an average of $80, compared to an average cost of $7,600 for correcting a vulnerability in production—slightly more than 1 percent of the cost of fixing a vulnerability in production.[6] The cost can be substantial. For example, as the average web application contains 26.7 vulnerabilities,[7] the cost of fixing these vulnerabilities in production would be over $202,920. And as most organizations have 10, 20, 30, or more applications in development, the costs can quickly spiral.

46%

of organizations had an incident caused by an unpatched vulnerability.[8]

[6] Eitan Worcel and Erez Rokah, "Integrate Application Security Testing into your SDLC," IBM, February 23, 2015.
[7] "Malware and ransomware attack volume down due to more targeted attacks," Help Net Security, February 5, 2020.
[8] "2020 CISO Benchmark Report: Securing What's Now and What's Next," Cisco, February 24, 2020..

03

Vulnerability Remediation Verification is Frustrating and Time-Consuming

Vulnerability Remediation Verification is Frustrating and Time-Consuming

While SAST and DAST can help to identify vulnerabilities in code, this is only part of the vulnerability management problem. Once a vulnerability has been identified, it needs to be **remediated** and **undergo additional testing** to ensure that the fix has occurred and does not create a new vulnerability or impact the functionality and security of the rest of the application.

In many cases, vulnerability remediation and remediation re-testing are manual processes. Once developers have run an application security test, which typically occurs near the end of the development life cycle, they are presented with a list of vulnerabilities to remediate. Manually remediating vulnerabilities, including identifying where the vulnerability exists in the code, and verifying that the security gap is closed takes up time, frustrates developers, and erodes relationships between the security and development teams.[9]

In response and in an attempt to meet release deadlines, security testing and vulnerability remediation are often partially completed or performed in a cursory manner. Any missed or unmitigated vulnerabilities could leave an organization open to attack through the application, creating security and compliance issues for the organization.

[9] "2019 Global Developer Report: DevSecOps," GitLab, July 2019.

04

Conclusion

As development processes accelerate to keep pace with the modern world of application development, organizations must adopt security automation in order to keep up while remaining secure. Traditional vulnerability detection solutions, such as SAST and DAST, which can take hours or days to analyze an application, are ill-suited to organizations making this move to DevSecOps.

One of the greatest limitations of these solutions is their lack of visibility into an application's internal operations. This limited visibility causes false positives and false negatives, which waste developers' time and create risk. A solution that "sits inside" an application and has full visibility into its state and execution paths can provide extremely high vulnerability detection accuracy and a very low false-positive rate.

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