CONTRAST SECURITY

PROTECTING APIS: AN UPHILL BATTLE

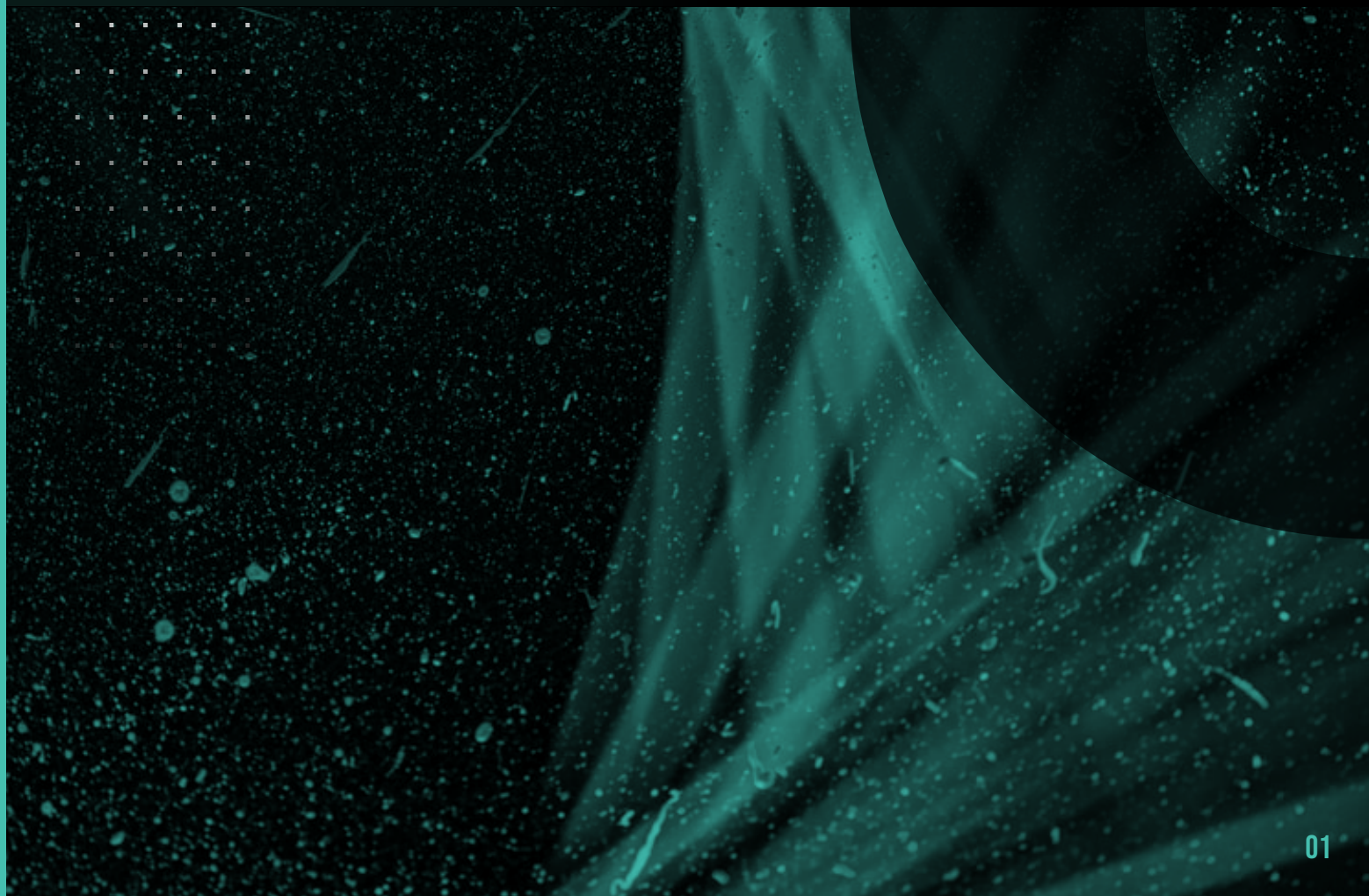
LEGACY APPLICATION SECURITY APPROACHES LEAVE GAPS IN PROTECTION

WHITE PAPER

EXECUTIVE SUMMARY

With applications an increasingly critical priority in organizations and the pace of development accelerating, application programming interfaces (APIs) are a vital part of every software application. Since APIs are the "connective tissue" that holds together the different parts of a piece of software, securing them is a critical priority for any organization. Unfortunately, too many companies struggle with application security in general, not to mention complex elements like APIs.

While most organizations include APIs in their regular security scans of software, legacy application security testing tools perform even more poorly with APIs than they do with standard code. There are at least three specific problems with traditional scans that impact API security: 1) a lack of continuous testing, 2) a lack of visibility into the routes actual users take in the software and data structures needed to test the API, and 3) a lack of actionable information to help prioritize the riskiest vulnerabilities for quicker remediation.



As digital transformation permeates the economy, organizations in all industries rely on software applications for virtually every aspect of operations, from sales to marketing and from product development to manufacturing. As one observer notes, "[Applications] have become *the* business imperative, *the* key conduit to customers, and *the* essential business enabler."[1]

Software engineers have responded to the increasing urgency to deliver applications by creating ways to develop them more quickly and efficiently. Methodologies like Agile and DevOps streamline processes and rely heavily on open-source code and microservices to prevent developers from "reinventing the wheel" for every element of a program.

APIS BRING IT ALL TOGETHER

APIs are, in many ways, the glue that holds these different elements together. More specifically, they allow different applications—or different elements within a single application—to communicate with each other within specified parameters. The use of API toolsets accelerates development cycles and introduces standardization into these interactions. Indeed, more and more applications are now built using an API-first design.

Internal APIs standardize the way that different elements of a program interact, helping "organize code and make components more reusable."[2] External or public APIs enable different pieces of software to work together, and facilitate user interactions with a web application. While APIs have been a tool in a developer's toolbox for decades, they are increasingly common for today's software architectures. One recent survey found that 67% of respondents expected to use APIs more frequently in 2020 than in 2019—and only 6% expected to use them less.[3]

STRUGGLES IN MANAGING API SECURITY RISKS

Unfortunately, APIs can also be a source of security risk for organizations.[4] Big-name brands like Venmo,[5] Facebook,[6] and Capital One[7] have had publicized API breaches in recent years, each affecting tens of millions of records.

Clearly, API security is a work in progress at many organizations. Recent research by Forrester[8] finds that organizations struggle to even assemble a complete and accurate inventory of all APIs in use—let alone ensure that they are free of vulnerabilities. One recent survey found that 88% of organizations use more than 500 APIs—and 63% use more than 1,000.[9] Keeping track of them can be a daunting task for security teams that are often already overwhelmed.

PROBLEMS WITH LEGACY TOOLS

APIs that are a part of software development are typically subjected to the same application security scanning tools and processes used for the rest of the application. However, legacy tools were not built for the pace and complexity of today's software development, and they have well-documented problems protecting the applications themselves.[10]

For APIs, traditional application security testing tools are arguably even *less* effective. This ineffectiveness can be broken down into three deficiencies: testing is not continuous, visibility into routes is not provided, and automated remediation prioritization is missing.

"[M]ISCONFIGURED OR OTHERWISE VULNERABLE APIS CAN AMOUNT TO THE DIGITAL VERSION OF UNLOCKED DOORS OR BROKEN WINDOWS, MAKING JUST ABOUT EVERY 'ROOM' IN THE HOUSE AVAILABLE."[11]

API TESTING IS NOT CONTINUOUS

At most organizations, APIs are included in the regular application security testing processes, which are anything but continuous. These scans are often done on a fairly regular basis. In fact, one major application security testing vendor found that vulnerability scans must be run *every day* in order to keep the mean time to remediation (MTTR) of vulnerabilities below 60 days.[12] But this frequency may not be possible at many organizations, due to insufficient application security staff or simply a decision to prioritize speedy development.

Even if a company is doing scans every day, the information from the scans is not available right away. Recent research found that application security testing scans take three hours or more at 91% of organizations—and eight hours or more at 45% of organizations.[13] But this is just the start of the process. Scan results are produced in a lengthy PDF report that must be triaged and diagnosed by application security specialists. Overall, the process of analyzing and triaging alerts takes an hour or more *per alert* at most companies.[14] Considering that many scan reports contain hundreds of alerts, this can tally into a huge time expenditure.

Of course, this wasted time also translates into a significant delay in getting actionable information to developers so they can perform remediation. By this time, the issue may be several days old. Additional

scans may have already been done, and developers have certainly moved on to subsequent steps of their work.

A lack of continuous scanning with real-time feedback complicates remediation because of the time lapse between a vulnerability being placed in an application and when developers are notified. And the later a vulnerability is discovered, the more expensive it is to fix.[15]

"[L]EGACY SECURITY TOOLS SIMPLY DON'T PERFORM WELL IN CHANGING ENVIRONMENTS LIKE THE CLOUD, CONTAINERS, OR MICROSERVICES."[16]

API TESTING DOES NOT BRING VISIBILITY INTO ROUTES

The purpose of developing software is not simply to amass lines of code. Rather, applications are meant to be used by human beings, who interact with them in specific ways. In modern applications, these interactions are often enabled and controlled by APIs. These data flows, or "routes," are also the ways that adversaries can find and exploit vulnerabilities.

Static vulnerability scanning has almost no visibility into how the software will actually be used, but rather looks for strings of code that might be a red flag for a vulnerability. Many vulnerabilities are not apparent until a user executes a particular route within the application. As a result, tools often require manual

engagement to understand how an API should be evaluated and how the public and private APIs connect. And even the vulnerabilities that are caught by a static scan may be more difficult to remediate because of a lack of background information about the route.

Dynamic scanning attempts to replicate a real user once an application has reached the stage where the software can be executed. But dynamic tools have trouble navigating the complexities of APIs, which makes it difficult for them to generate well-formed requests for testing.[17] Specifically, manual work often must be done to analyze what data structures and values will cause the API to execute a valid condition. The result is that this attempt at runtime analysis may miss vulnerabilities in APIs altogether.

A lack of visibility into routes creates one additional problem: alerts about vulnerabilities that pose no risk to an organization. When open-source content—including APIs—is incorporated into a piece of software, there may be extraneous portions of that code that are not used by the software. If a vulnerability does not lie on any of the possible routes that a user could take through the application, it poses no risk to an organization. A lack of visibility into routes results in wasted developer time with unnecessary remediation.

"[S]TATIC TESTING METHODS CANNOT PROVIDE A CLEAR PICTURE OR COMPLETE VISIBILITY OF THE APPLICATION ATTACK SURFACE."[18]

API TESTING DOES NOT SUPPORT PRIORITIZATION OF REMEDIATION

While every organization should have a goal of remediating every vulnerability that poses any risk at all, the sheer number of vulnerabilities means that prioritization is the key to minimizing risk. Research by Contrast Labs consistently shows that between 97% and 99% of applications in development have at least one vulnerability, and those that have any vulnerabilities average 50-plus per application.[19] With these numbers, it is important to have an idea of the relative risk posed by each vulnerability.

This is a big drawback to legacy application security testing tools, whether for regular code or for APIs. Static and dynamic scans simply provide lists of alerts. Once the large number of false positives are weeded out, security teams are left to figure out which vulnerabilities should be addressed first.

One resource for open-source vulnerabilities is the Common Vulnerability Scoring System (CVSS), used to rate the severity of entries in the Common Vulnerabilities and Exposures (CVE) database. However, these scores are general and do not take into account the specific needs of an organization—or even what industry that company is a part of. And even this resource is not available for vulnerabilities that occur in custom code. In the end, staff can spend so much time analyzing the risk posed by each vulnerability that little progress is made on actual remediation.

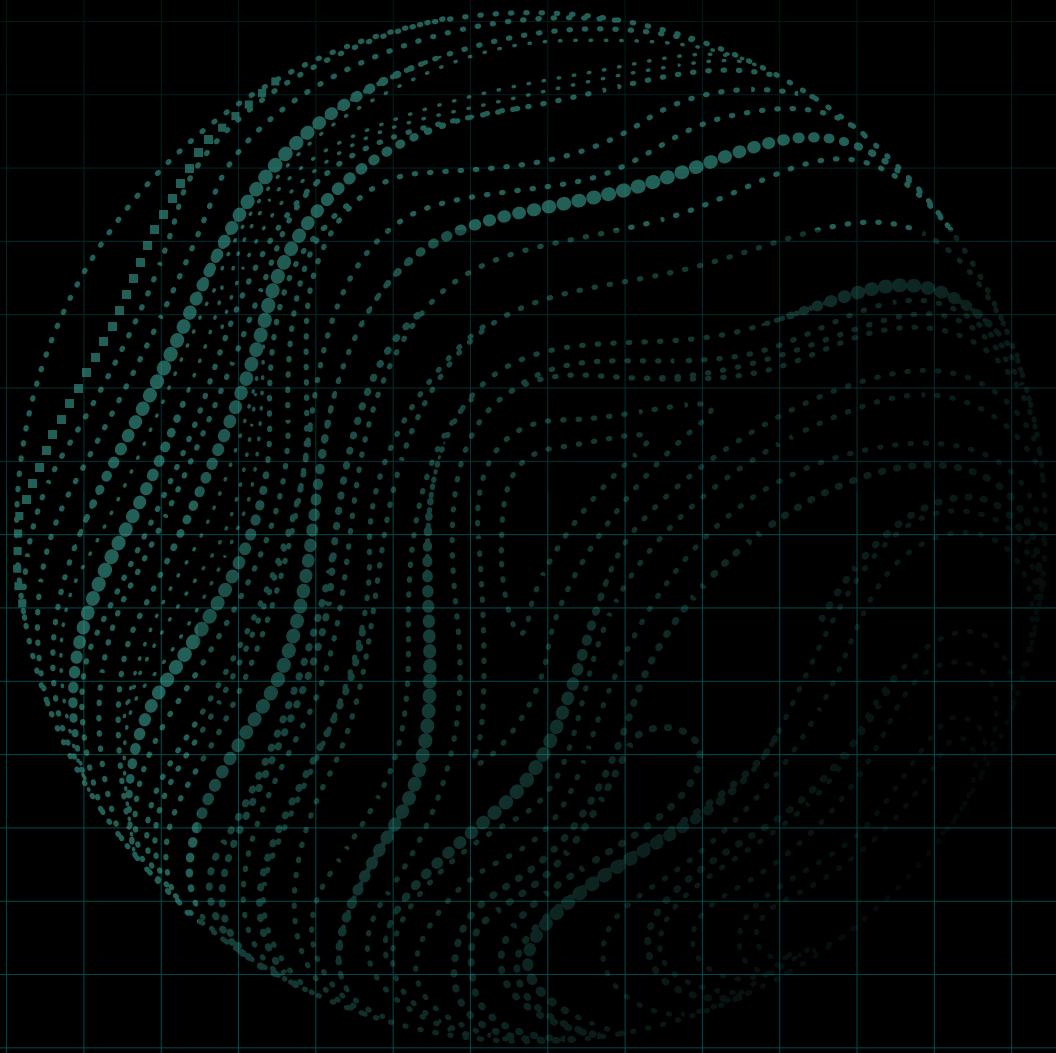
"AS VULNERABILITIES ARE REMEDIATED FASTER—ESPECIALLY THOSE THAT ARE SERIOUS—THE RISK POSED BY THE APPLICATION ATTACK SURFACE DIMINISHES."[20]

CONCLUSION

With the number of APIs found in many organizations' roster of applications ballooning into the hundreds or thousands, it is critical for organizations to find a workable way to identify vulnerabilities in a timely manner, prioritize them for remediation, and ensure that no API vulnerabilities are present when an application goes into production. This goal of "shifting left"[21] is complicated by a lack of continuous scanning, missing route visibility, and no automated way to prioritize vulnerabilities.

As the pace of digital transformation continues to increase, APIs will likely grow in number and importance at most organizations in the coming years. This will only complicate and exacerbate the problems discussed in this white paper. The result will likely be more well-publicized breaches that are traced to poorly protected APIs—at widely recognized brands and obscure ones alike. A better solution is needed—one that provides real-time vulnerability feedback based on the ways users will interact with the software.

[1] Jo Peterson, “DevOps: The Secret to Doing More, Faster, Better and for Less Green,” Channel Futures, February 23, 2018.
[2] Jonathan Freeman, “What is an API? Application programming interfaces explained,” InfoWorld, August 8, 2019.
[3] Tom Donoghue, “API Adoption To Increase in 2020; REST, Serverless and GraphQL Most Popular Survey Says,” Integration Developer News,” accessed November 28, 2020.
[4] Lucy Kerner, “Critical API security risks: 10 best practices,” TechBeacon, August 4, 2020.
[5] Dan Salmon, “I Scraped Millions of Venmo Payments. Your Data Is at Risk,” WIRED, June 26, 2019.
[6] Mike Isaac and Sheera Frenkel, “Facebook Security Breach Exposes Accounts of 50 Million Users,” The New York Times, September 28, 2018.
[7] Lily Hay Newman, “Everything We Know About the Capital One Hacking Case So Far,” WIRED, August 29, 2019.
[8] Sandy Carielli, et al., “API Insecurity: The Lurking Threat In Your Software,” Forrester, October 22, 2020.
[9] “The State of DevSecOps Report,” Contrast Security, December 2020.
[10] “A Major Roadblock To Business Innovation: How Traditional AppSec Delays DevOps Release Cycles," Contrast Security, April 14, 2020.
[11] Taylor Armerding, “It’s Past Time To Pay Much More Attention To API Security,” Forbes, December 5, 2018.
[12] “State of Software Security, Volume 11,” Veracode, October 27, 2020.
[13] “The State of DevSecOps Report,” Contrast Security, December 2020.
[14] “Ibid.
[15] Mukesh Soni, “Defect Prevention: Reducing Costs and Enhancing Quality,” iSixSigma, accessed April 9, 2020.
[16] Patrick Spencer, “Traditional AppSec Code Halts Kill DevOps Release Cycles,” Security Boulevard, April 30, 2020.
[17] Jeff Williams, “What Do You Mean My Security Tools Don’t Work on APIs?!!” Dark Reading, June 25, 2015.
[18] Subhash Arja, “Route Intelligence™ Enables Transformation of Traditional Application Security Testing,” Contrast Security, March 17, 2020.
[19] “Contrast Labs Application Security Intelligence Report, September–October 2020,” Contrast Security, December 2020.
[20] Katharine Watson, “Application Risk Is 1.7x Higher for Organizations That Fail To Manage Security Debt,” Contrast Security, July 24, 2020.
[21] Jakob Pennington, “Shifting Left: DevSecOps as an Approach to Building Secure Applications,” Medium, July 18, 2019.



CONTRAST SECURITY

240 3rd Street
Los Altos, CA 94022
888.371.1333

Contrast Security is the world's leading provider of security technology that enables software applications to protect themselves against cyberattacks, heralding the new era of self-protecting software. Contrast's patented deep security instrumentation is the breakthrough technology that enables highly accurate assessment and always-on protection of an entire application portfolio, without disruptive scanning or expensive security experts. Only Contrast has sensors that work actively inside applications to uncover vulnerabilities, prevent data breaches, and secure the entire enterprise from development, to operations, to production.