noname

The D.A.R.T. Strategy for API Security

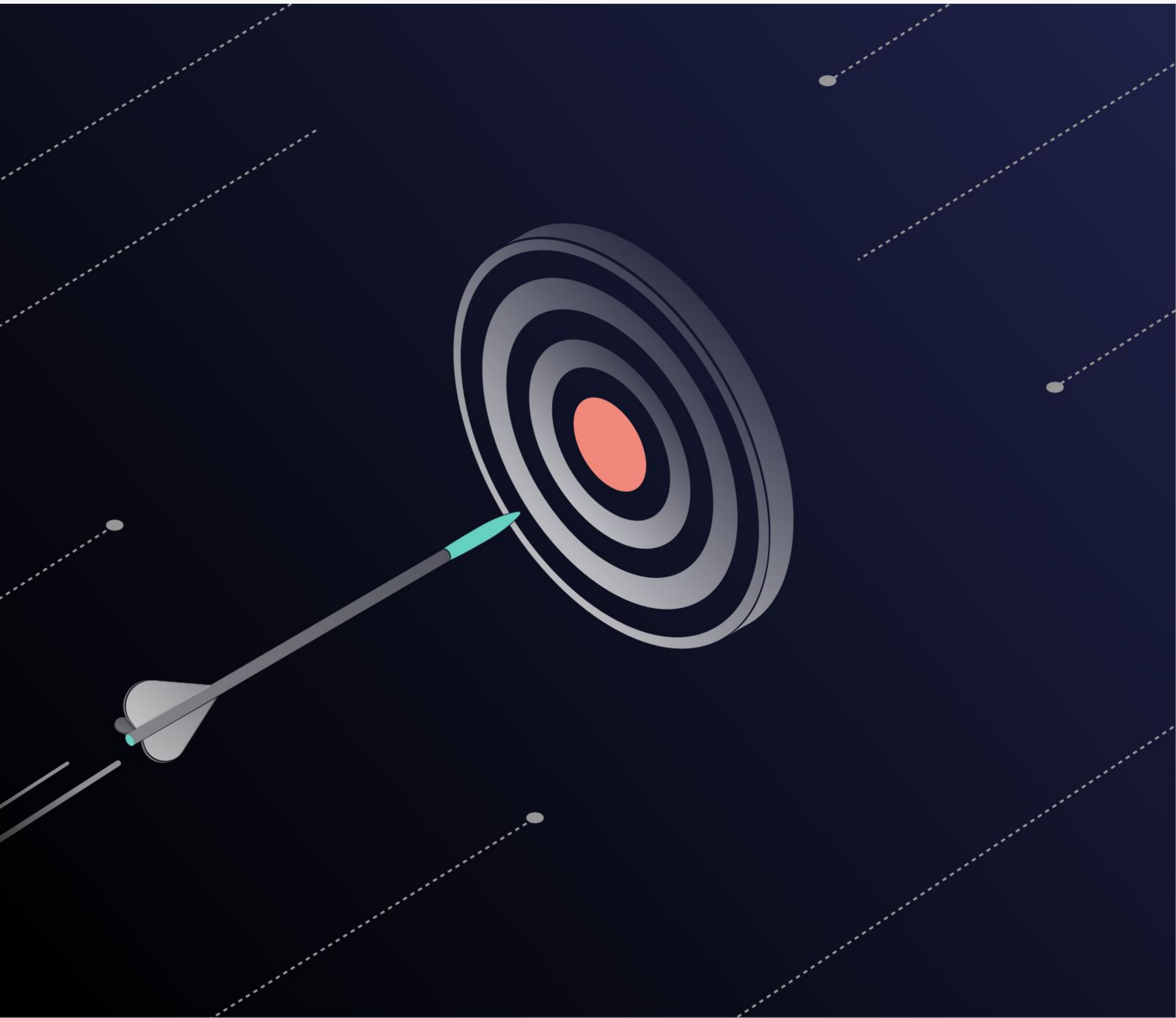
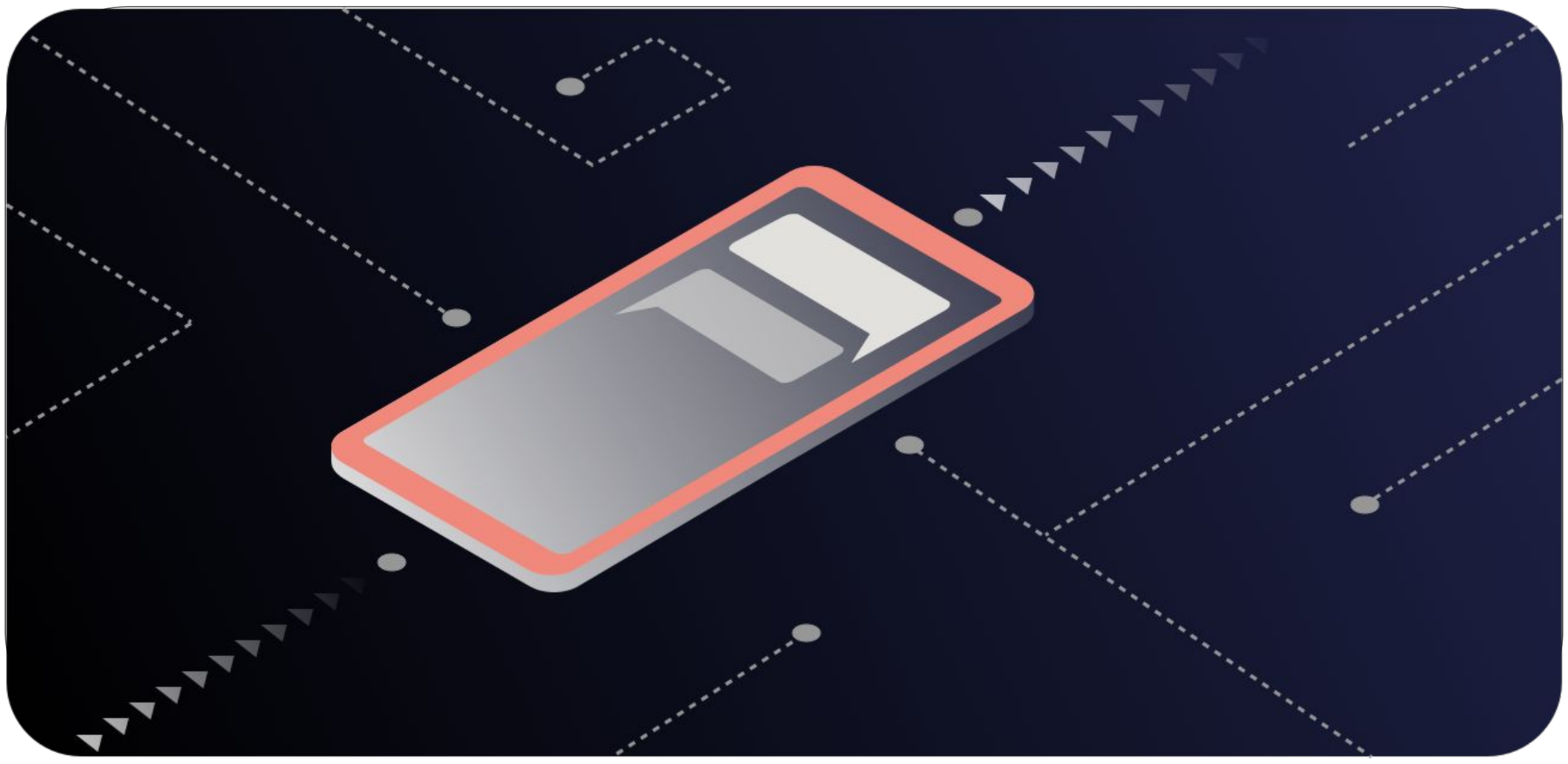


Table of contents



Introduction

Modern applications are highly interconnected. Just as the internet dramatically increased the value of a single computer, distributed applications have created far more powerful and useful software. These applications interoperate through documented Application Programmable Interfaces (APIs) coded to open their functions and data to other applications. It’s how you get the refund in your bank account for the shoes you returned to an online retailer or how you get added to the payroll system at your new employer during onboarding. Events in one application drive data updates and actions in adjacent apps - updating records, triggering notifications, starting or stopping processes.

The adoption of cloud-native, distributed applications has accelerated reliance on APIs. Today, by some estimates, API calls represent 83% of all web traffic. Since they provide direct access to critical services and data, APIs have become a rich target for hackers.

Gartner estimates that in 2021, APIs will account for 90% of the attack surface for web-enabled applications, and API abuse will become the most frequent attack vector by 2022.

Modern application and API management are multi-layered involving technologies such as Web Application Firewalls (WAFs), Application Delivery Controllers (ADCs) and load balancers, anti-virus and signature engines, intrusion and anomaly detection, and API gateways, among others. The challenge here is that most of these solutions don't provide a standardized way to inventory, monitor, and secure APIs. For example, the API gateways will only monitor known APIs.

Many APIs exist outside this universe of central control either because they weren't adequately documented, weren't decommissioned, were poorly configured, or even maliciously left open.

As enterprises continue their digital transformation journey, it is critical for security leaders to take into account how the modernization of their infrastructure, applications, and services necessitates a modernization of API management and security. Just as the definition of cloud security and application security continues to shift in significance and scope, API security needs to be re-defined for a modernizing world. **Noname Security defines API security as the following:**

API security refers to protecting the integrity of your digital environment from API vulnerabilities, API misconfigurations, and API cyber attacks.

This broader definition is required for cloud-first and API-led organizations. While it is important to protect APIs from the threat of outside attacks, equally as important is the need to identify and resolve human errors, leaks, and design errors that can inadvertently expose sensitive information or provide an unintended attack surface that could be exploited.

Modern API security requires a more holistic approach called D.A.R.T.:

Discover

find all APIs across the environment, including rogue and shadow APIs

Analyze

detect API attacks, suspicious behavior, and misconfigurations

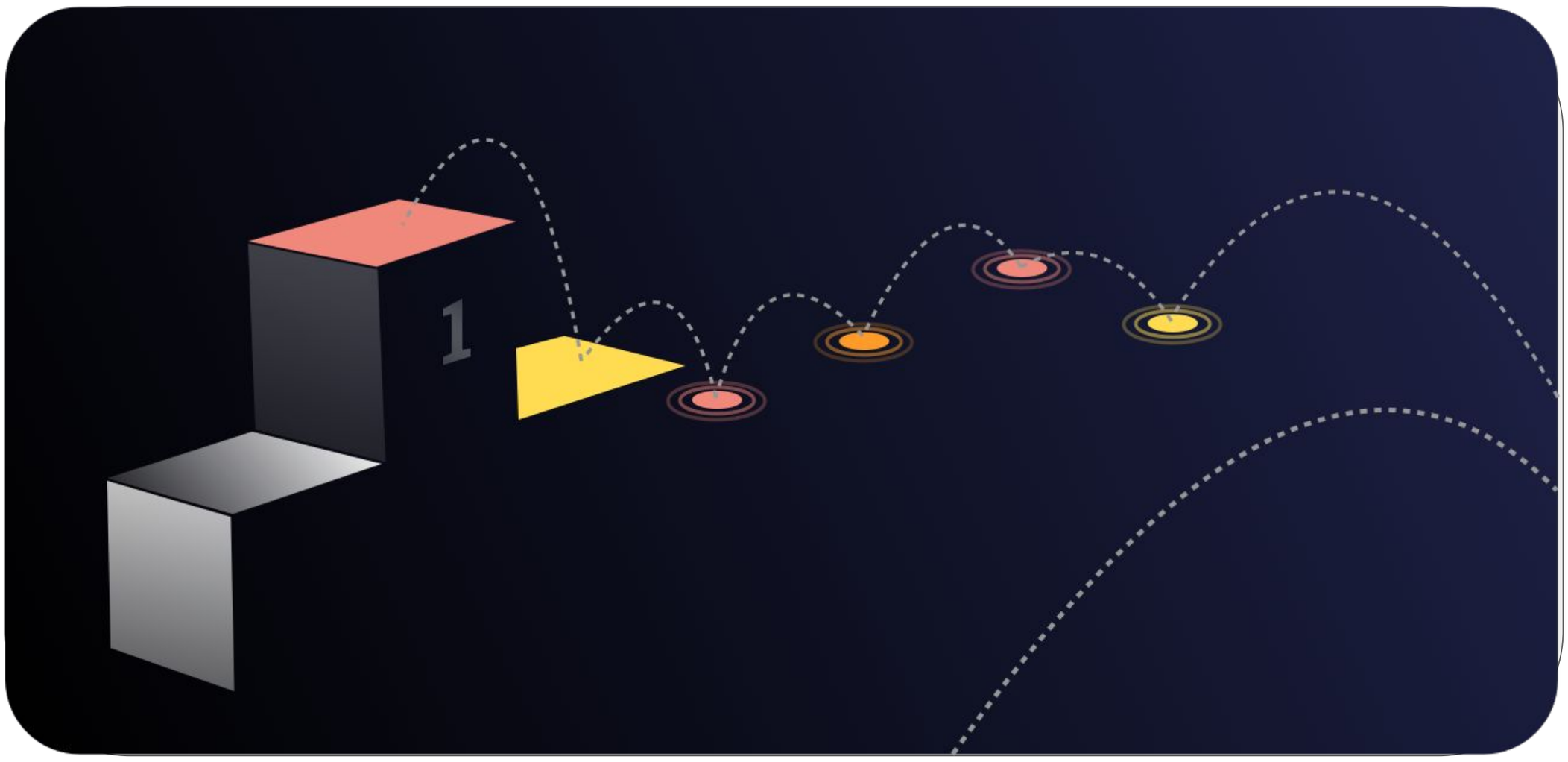
Remediate

prevent security breaches and data loss, and integrate with existing management solutions

Test

actively test APIs before production and after deployment

This guide will introduce the D.A.R.T. strategy and explain how an API security platform should support it.



The Rise and Risk of APIs

Before we dive in, let’s start with a bit of context - application networking is nothing new. There is a long history of interconnecting software applications.

Service-Oriented Architectures aim to create distributed, distinct but interoperable components of an application accessible over a network. This approach relies on standards for exchanging information, including CORBA (Common Object Request Broker Architecture) designed in the early 1990s, SOAP (Simple Object Access Protocol) from the late 1990s, to other standards in common use today, including REST (Representational State Transfer) first codified in the early 2000s up to GraphQL (an open-source data query language), which Facebook first developed in 2012.

All these efforts aim to enhance the power and interoperability of applications over the Web. As organizations push workloads to the public cloud, adopt microservices, and transition to digital propositions - the use of these technologies and APIs has exploded.

It's no exaggeration to say that most modern organizations have no idea just how many APIs are in use in their organizations today.

Organizational reliance on APIs has made them a significant target for malicious actors. As long ago as 2017, the Open Web Application Security Project, aka the OWASP Foundation, produced its Top Ten list of web application security exploits

In 2019, OWASP published a dedicated API Security Top Ten list. These remain the primary security risks developers should be aware of, including code injection; broken authentication; data exposure; poor XML configuration; broken access controls; security misconfigurations; cross-site scripting; insecure deserialization that allows remote code execution; use of vulnerable components; and insufficient monitoring.

Even though the OWASP Top 10 has been widely publicized since 2017 and the OWASP API Security Top 10 has been out since 2019, enterprises make headlines every week when their efforts fail to protect them against these and other API security exploits.

The Three Ms of API Security

Why are well-resourced and sophisticated organizations falling victim to well-documented exploits?

The answer lies in the three Ms of API Security - and sadly, many organizations are ill-suited to detect or prevent them.

1 Mistake - errors in the API lifecycle

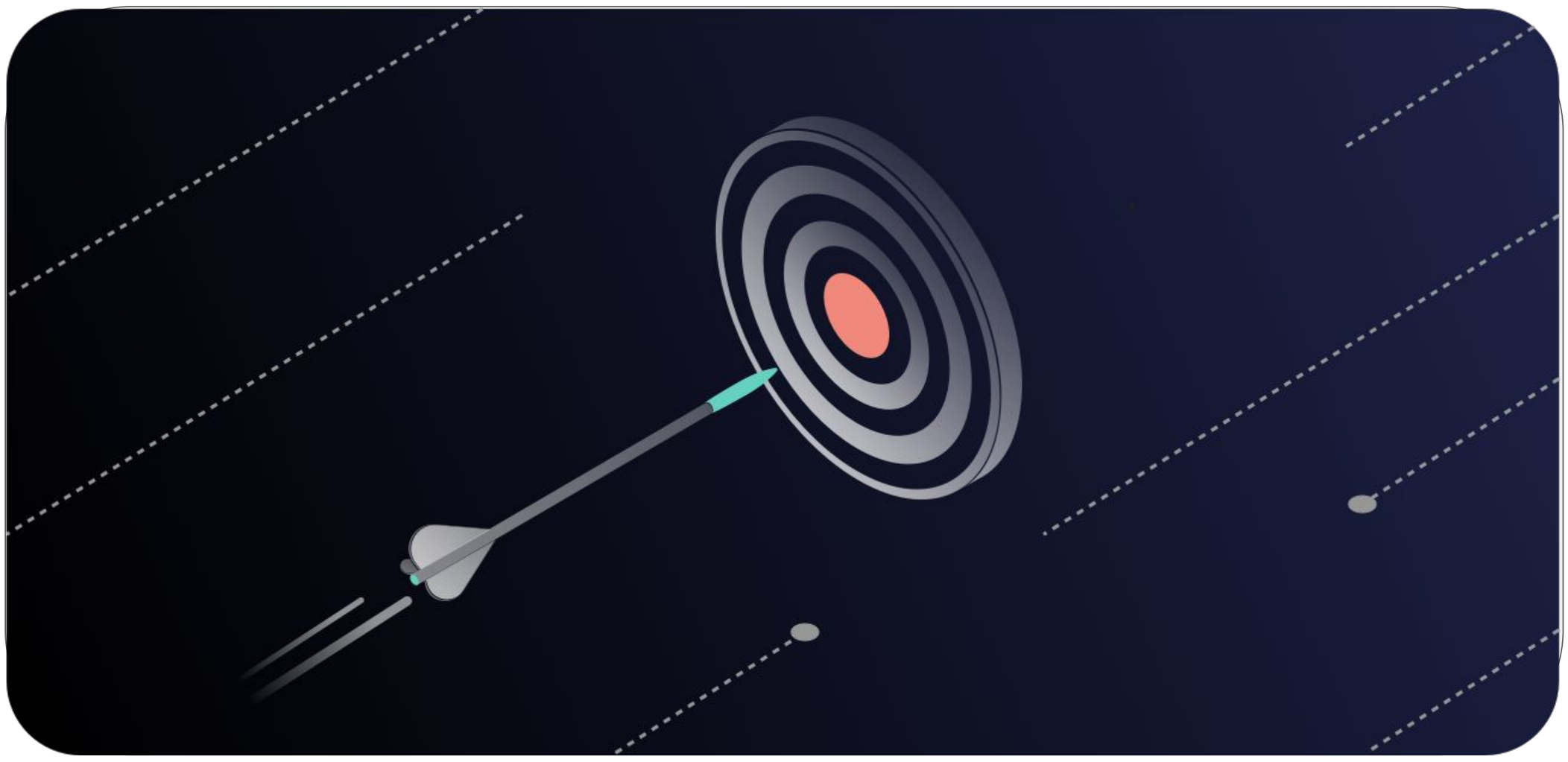
The first cause of API security flaws is simple human error. APIs are just as complex to develop and maintain as the applications they connect. As a result, it is easy to misconfigure policies, to host multiple versions of an API, and for temporary APIs to be left open by time-pressed DevOps teams.

2 Misfortune - unforeseen events that create exposure

The second source of API vulnerabilities is simply bad luck. An unforeseen sequence of events can expose an API to the open web or allow unauthorized access. As the use of APIs explodes, such mishaps will likely occur with increasing frequency. All companies know that things will go wrong - and it's incumbent upon them to quickly detect those failures and close the security gaps they create.

3 Mischief - deliberate exploit of the API to disrupt or exfiltrate data

The third type of vulnerability comes from a deliberate attack on the API itself. It could originate from outside the organization or within. By their nature, APIs are well documented and version-controlled. It's evident how the API should respond, so a hacker can look to exploit it armed with that information. For instance, malicious code designed to subvert or disrupt the service can be injected into a feature allowing a user to upload a profile photo. Or an authentication challenge might be attacked through credential stuffing overwhelming the service. Some versions of standard APIs are vulnerable to specific attack vectors, and hackers will probe them to see whether those exploits are live before launching their attacks. In addition, bad actors are looking to exploit deployment and configuration mistakes made by organizations and take advantage of their misfortune when the opportunity strikes.



The D.A.R.T. API Security Strategy

Overview of D.A.R.T.

Given the increased reliance on APIs, their importance to digital businesses, and the rising level of sophistication of hackers looking to compromise those APIs, organizations need a proven strategy for API security. It must cover all the stages of the API lifecycle and support modern application development processes. We call this strategy D.A.R.T. - Discover, Analyze, Remediate, and Test. While the Noname API Security Platform is designed around this approach, any organization can adopt the strategy to ensure a positive API security stance.

The premise is simple - first, you need to Discover all the APIs operating within the organization; then Analyze them to see their configuration and to assess their behavior; Remediate any active vulnerabilities; and finally, Test new software or any updates to ensure continued security compliance. This D.A.R.T. process is a constant and interrelated cycle that will safeguard the company's systems, data, and reputation.

Let's take a deeper look at each of the components of the D.A.R.T. strategy.

Discover

Discover refers to the ability to find and inventory all APIs and their associated rich data.

Enterprises manage thousands of APIs, many of which are not routed through a proxy such as an API Gateway or WAF. This means they are not monitored, rarely audited, and are most vulnerable to mistakes, misfortune, and mischief. There are several ways to discover a complete inventory of APIs, some more resource-intensive than others. However, it's most important to create a comprehensive API inventory to prevent unseen vulnerabilities.

For every API, you need to know what type of sensitive data it is interacting with, who the owner is, how it is routed (internally or externally), the associated physical resource (such as instance-id and server name), and which application or business unit it belongs to. This information can be gathered from the API header and body. The header refers to meta-data associated with the delivery of API requests and responses. The body refers to the data being sent to and from the API.

An effective API security platform discovers all of this information, so you have the visibility and understanding necessary to analyze and remediate potential vulnerabilities. Ideally, this would not require installing agents, modifying the network, or any other changes that might impact application performance. Historically, this has been an issue with API discovery - not only is it time-consuming, but it causes service degradation.

For example, collecting API data through agents requires them to be installed, configured, and maintained. The additional effort only becomes more complex as enterprises scale API usage. Additionally, and more importantly, agents sit in-line and are only effective at discovering APIs that you know exist. Agent-based solutions can't find legacy and shadow APIs that aren't actively managed by a gateway or WAF. It's also important to note that discovery isn't a one-off process - it needs to be conducted repeatedly to capture any changes or new vulnerabilities introduced to the network.

Once the API header and body information is captured, it needs to be organized and presented in an actionable and clear format. The data, such as the owner of the API, the API version, level of use, whether internal or external, needs to be presented to give a clear picture of the current state of the API environment. From here, we move on to the second stage of the D.A.R.T. API Security Strategy - Analyze.

Analyze

> Analyze refers to the ability to detect API attacks, anomalies, and misconfigurations.

Enterprises need to understand API access, usage, and behavior. However, APIs are complex to analyze - it’s not as simple as searching for abnormal behavior in API traffic in the way a traditional SIEM or log management system might. A monitor-and-alert model like that would produce too many false positives as each unexpected call would trigger a warning. It can quickly overwhelm a security team or simply be ignored as noise. Equally, just looking at the network traffic will not provide a deep understanding of API behavior or the reasons behind it.

Understanding the API security landscape in all its complexity requires processes such as parsing logs, ingesting catalog data, reviewing configurations, security testing, and assessing device configurations. The problem is many organizations have already developed and matured their agile CI/CD or SDLC processes. Trying to force security into those processes is likely to create internal friction and delay. Even organizations that have built security standards and compliance into their CI/CD processes will occasionally need to make sacrifices for the sake of business requirements. Shortcuts lead to inconsistencies in API security, and hence vulnerabilities.

Shift-left testing to find and resolve issues early on makes sense. But despite this, and even with skilled network architects, identity managers, and cybersecurity architects, APIs are deployed every day with significant security issues. Only by thoroughly analyzing the API ecosystem can we identify serious problems such as misconfigured, misrouted, poorly authenticated, and unauthorized APIs.

The analysis process must be automated, repeatable, and actionable. It needs to identify issues so they can be remediated before they can be exploited. Delays in analysis provide additional time for hackers to take advantage of the vulnerabilities. A proactive strategy exposes vulnerabilities, misconfiguration issues, and changes before they can be exploited and provides an effective window for remediation to occur so the attack can't happen in the first place.

This leads us to the next stage in the API Security Strategy - Remediate.

Remediate

> Remediate refers to the ability to prevent attacks and resolve misconfigurations.

There are several approaches to resolving API security issues, including blocking API attacks in real-time and integrating with existing remediation workflows and security infrastructure. However, the team must get the information and alerts it needs to react immediately.

Without the proper tools, remediation can be complex, either because it is technically challenging or because it requires considerable time and effort. However, if remediation is done right, it can often be automated or semi-automated and, after that require little or no human interaction.

The remediation process should be integrated with the company's existing IT workflow management system (e.g., Jira, ServiceNow, or Slack). This means issues can automatically be assigned to the appropriate teams as they are identified

without changing workflow or a different system to check. Typically, the security team needs to work with IT, DevOps, or the business unit to remediate the threat, so automatically assigning them to the right group within a familiar system significantly reduces the workload.

For example, let's imagine an organization has a mobile app that resides in the /mobile subdomain. Routine analysis identifies a new configuration issue within the mobile app, a ticket is created in the IT management system, and the appropriate product manager is immediately notified. The product manager can then assign the ticket to the appropriate developer or DevOps team. Depending on the severity of the issue, the fix can be scheduled for the next sprint, or an emergency patch can handle it.

If the product manager sees this kind of ticket regularly, logic can be added to send the ticket directly to the correct team (or team member) with a copy going to the product manager.

While automation will accelerate remediation, it's best practice initially for a human to approve an action before it is taken. This will prevent services from being unexpectedly discontinued. For example, analysis reveals that an API exposes too much data, but the API should not automatically be blocked because its value to the business unit is unknown. With coordination from the business unit, semi-automated remediation can temporarily block the API use.

Over time, some issues are more likely to recur than others. These are the issues that should be considered for automated remediation. For example, an API that does not have the appropriate security policy could be identified and have it added automatically. This issue can be resolved in seconds and with no interruption of service or risk to the business unit, leaving an API that is more secure and resistant to attack, lowering the organization's risk posture.

An effective remediation process should also identify and prevent exploits as they occur. Not every vulnerability can be addressed before someone attempts to exploit it. However, attackers must take steps to actively exploit an API, and those may display their intent before they can complete their attack. For instance, a hacker will often try to evade the WAF, perhaps through directory traversal. No legitimate API traffic will deliberately avoid the WAF, exposing the attacker's user profile (if authenticated) and IP address. This means automated or semi-automated remediation techniques can block the attacker at the firewall or the WAF or revoke their credentials.

The process of discovering valuable APIs requires making multiple unauthorized calls that should result in 404 or 403 errors. This anomalous activity can be detected and alerts sent to the security team to activate automated or semi-automated remediation to block the user. An adversary will often recognize when they are blocked and refrain from making further attempts in search of another target.

Equally repeated calls to the same API or an API transmitting more information than expected to a single user should be identified as anomalous behavior. At this point, more automated or semi-automated responses can be activated to stop the attacker and mitigate future attacks.

Test

Test refers to actively testing your APIs to validate integrity before and after they are deployed to production.

Many of the API security issues you've seen in the news could have been avoided entirely if thorough testing had been applied. In other words, you need to analyze your APIs and remediate issues while in development. This allows you to deploy APIs at the speed of your business with complete confidence and trust.

As most organizations have embraced agile development and maintaining a CI/CD pipeline, APIs are often overlooked. While the applications undergo rigorous iteration and testing, APIs do not. Enterprises need to adopt a suite of tests specific to APIs to identify misconfigurations and vulnerabilities before the APIs are used in production.

Additionally, and perhaps more important, is the need for ongoing, active testing of all APIs already in production. In most modern environments, the applications and services, the underlying infrastructure, and the APIs themselves are dynamic. There is a constant change in traffic, policies, and functionality of the digital environment, which means that the security posture is also experiencing constant change. Just because an API was configured correctly at deployment doesn't mean that the API is impervious to misconfigurations in the future.

APIs change with the needs of the business and it is critical that the IT and security teams are immediately alerted when production APIs start to fail active tests. API security shouldn't be limited to a moment in time, and active API testing ensures that the integrity of APIs is maintained throughout the entire software development lifecycle.

Conclusion

APIs are critical to modern digital organizations providing access to data and services within the organization and with partners. Unfortunately, they are also a rich target for malicious exploitation whose victims include some of the largest companies across sectors from financial services to digital entertainment. The commercial, financial, compliance and reputational risks are clear and increasing.

Yet, most organizations have no cohesive approach to API security. They don't have a complete inventory of active APIs, they aren't documented, haven't been assessed for vulnerabilities, and protection measures are ad hoc or even absent.

The D.A.R.T. API Security Methodology provides a framework for organizations to mitigate this risk. Independent of any specific platform, it lays out the critical steps for enterprises to take to implement a proactive API security strategy:

Discover

detect all APIs across the network, including rogue and shadow APIs

Analyze

understand API behaviors and configurations

Remediate

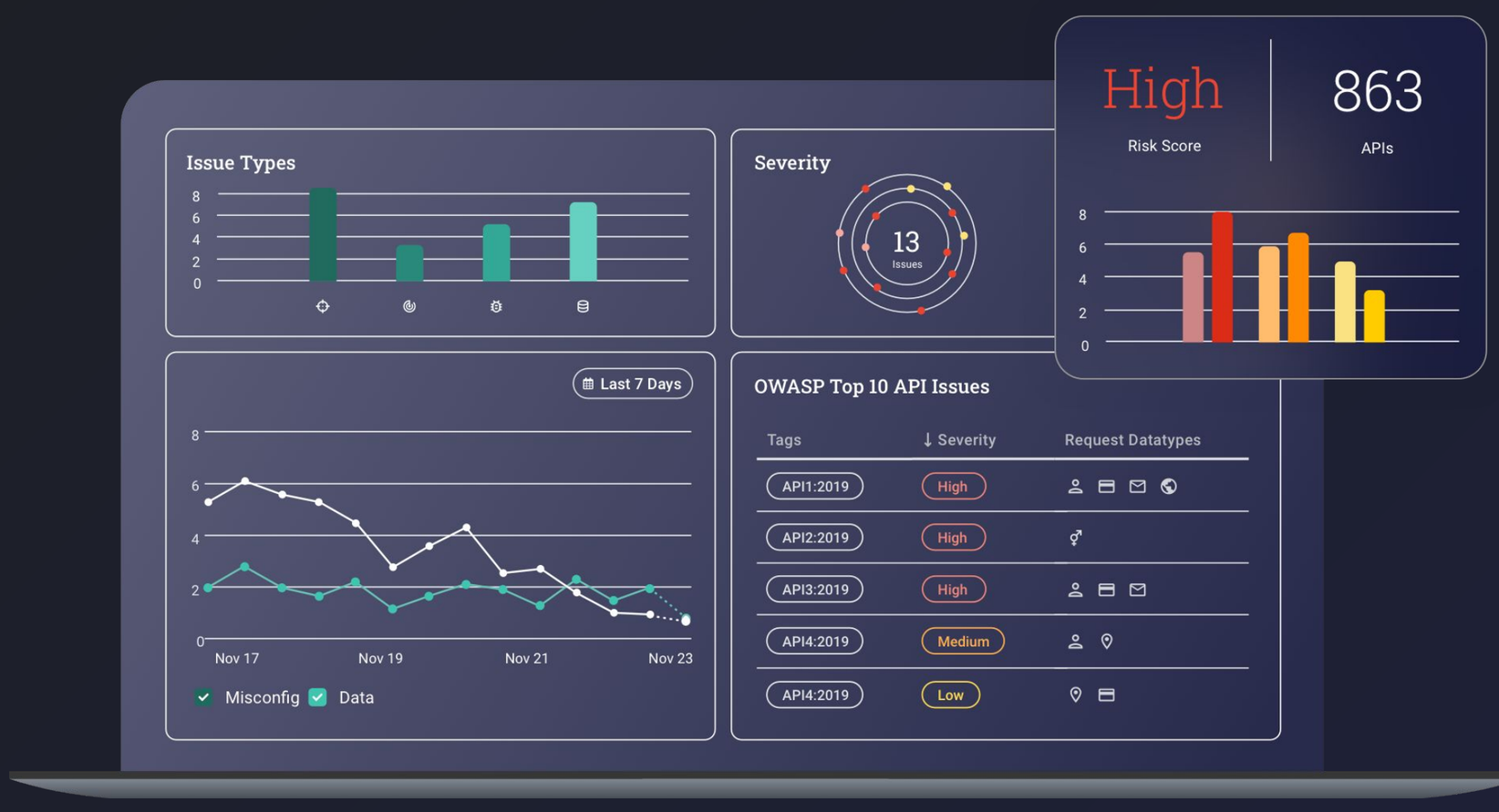
take measures to prevent security breaches and data loss

Test

ensure those steps are successful in preventing common API attacks

D.A.R.T. represents a comprehensive, interlocked, and repeatable process. Each stage will help improve the organization's security posture and provide clear next steps to close vulnerabilities. While the API security risk is rising, the D.A.R.T. methodology provides an approach for organizations to meet it, not just for today but also for tomorrow.

noname



About Noname Security

Noname Security creates the most powerful, complete, and easy-to-use API security platform that helps enterprises discover, analyze, remediate, and test all legacy and modern APIs. Noname finds and inventories all APIs using traffic mirroring, leverages AI-based anomaly detection and behavioral analysis, and integrates with existing remediation workflows and security infrastructure.

Get a Demo