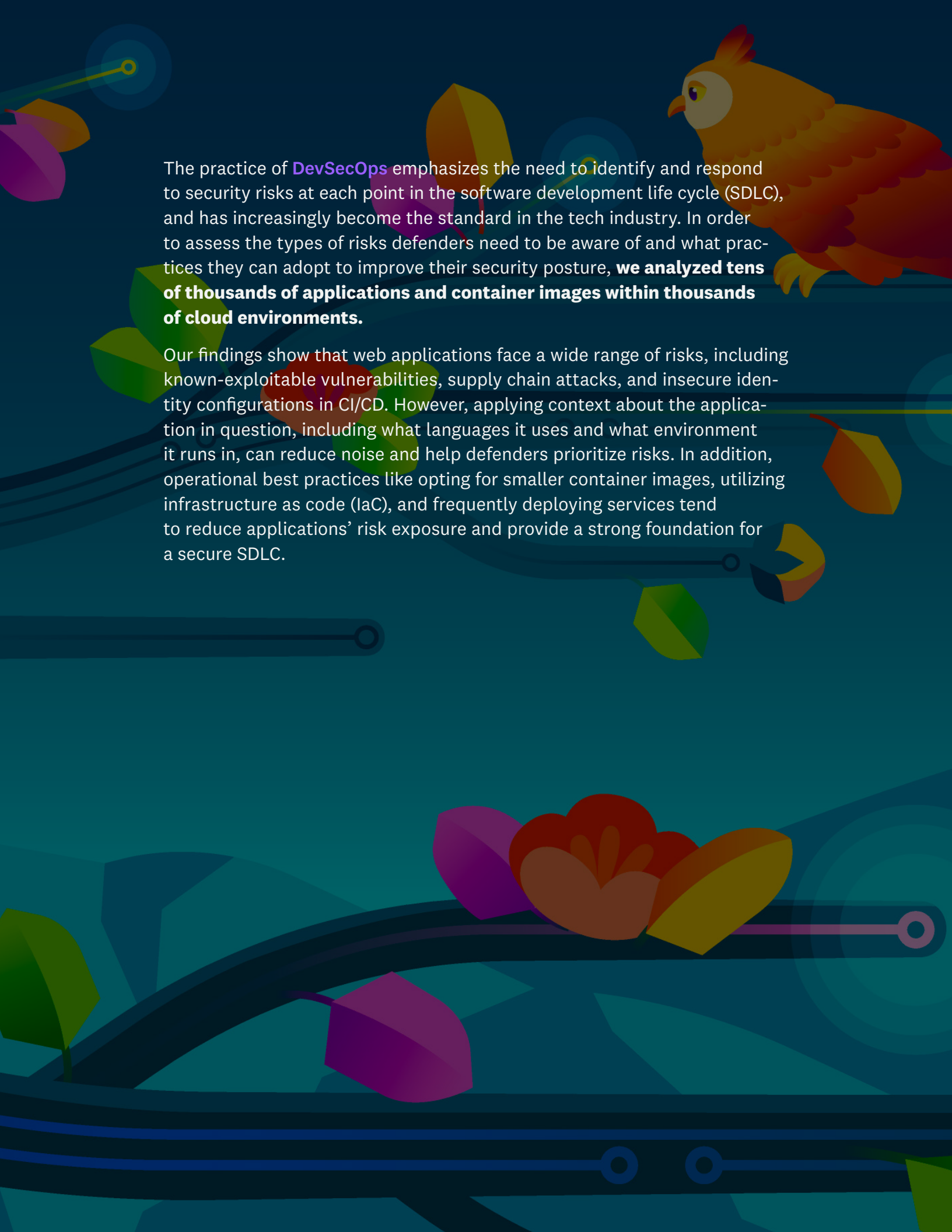
STATE OF DevSecOps

DATADOG



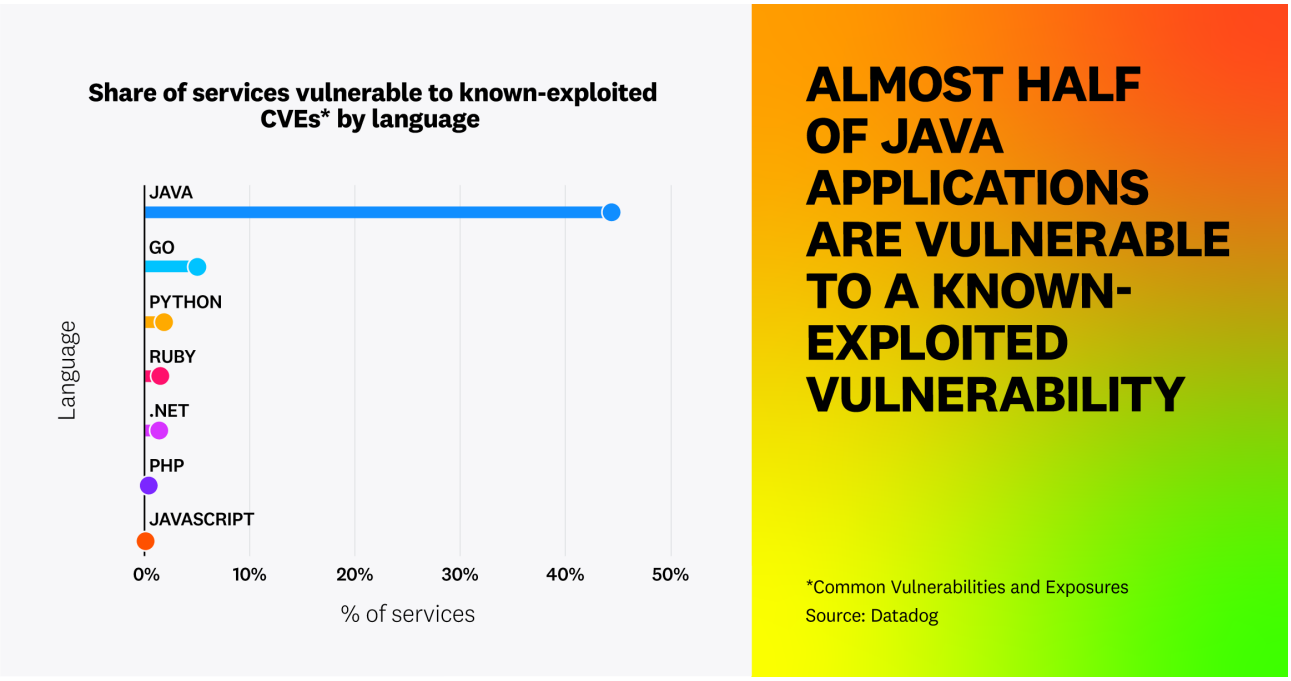
The practice of DevSecOps emphasizes the need to identify and respond to security risks at each point in the software development life cycle (SDLC), and has increasingly become the standard in the tech industry. In order to assess the types of risks defenders need to be aware of and what practices they can adopt to improve their security posture, **we analyzed tens of thousands of applications and container images within thousands of cloud environments.**

Our findings show that web applications face a wide range of risks, including known-exploitable vulnerabilities, supply chain attacks, and insecure identity configurations in CI/CD. However, applying context about the application in question, including what languages it uses and what environment it runs in, can reduce noise and help defenders prioritize risks. In addition, operational best practices like opting for smaller container images, utilizing infrastructure as code (IaC), and frequently deploying services tend to reduce applications' risk exposure and provide a strong foundation for a secure SDLC.

FACT 1

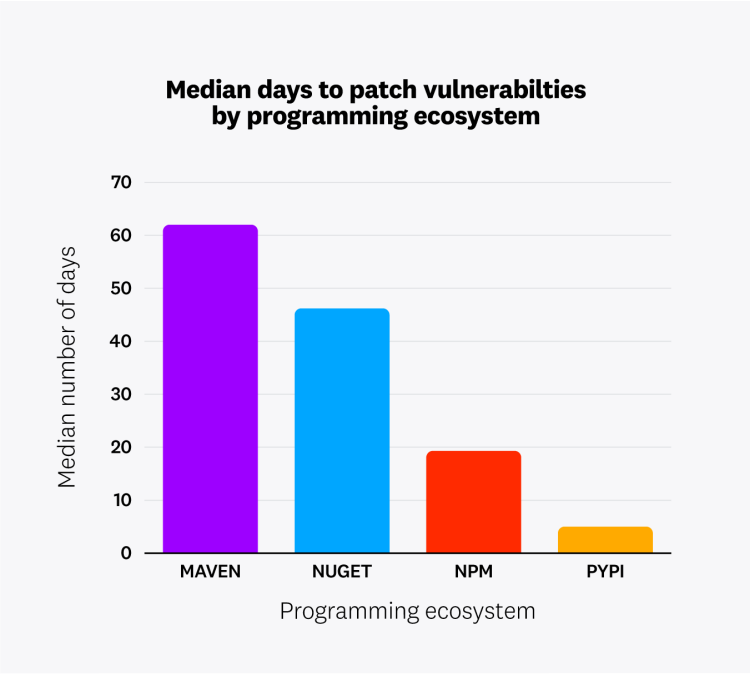
Exploitable vulnerabilities are prevalent in web applications, particularly those that use Java

By analyzing our dataset of applications, we found that **15 percent of services are vulnerable to known-exploited vulnerabilities,** affecting 30 percent of organizations. Vulnerabilities are particularly prevalent among Java services, **44 percent** of which contain a known-exploited vulnerability. The next highest is Go at 5 percent, with the average for all non-Java services at 2 percent.



In fact, 14 percent of Java services still contain at least one vulnerability even when we look at only those that are known to be highly impactful, such as known remote code exploitation (RCE) vulnerabilities like Log4Shell, Spring4Shell, and other routinely exploited attack paths.

In addition to being more likely to contain high-impact vulnerabilities, Java applications are also patched more slowly than those from other programming ecosystems. We found that applications from the Java-based Apache Maven ecosystem had a median time to fix library vulnerabilities of **62 days**, compared to 46 days for those in the .NET-based NuGet ecosystem and 19 days for applications built using npm packages, which are JavaScript-based.



TIME TO PATCH IS LONGEST FOR VULNERABILITIES IN THE JAVA-BASED MAVEN ECOSYSTEM

Source: Datadog

Leaving known vulnerabilities unpatched for prolonged periods is particularly risky because web applications are often exposed to the internet. Attackers use automated scanners to continuously scan the internet for high-impact, easy-to-exploit vulnerabilities, and recent research shows that vulnerabilities are often exploited just hours after they are initially disclosed.

Our research confirms this pattern of attacker behavior. We found that **88 percent of organizations received untargeted malicious HTTP requests,** such as to `/backup.sql`, scanning for potentially exposed sensitive files or API routes. In addition, **65 percent of cases where a specific attacker attempted to exploit a specific URL were untargeted**—i.e., the same attacker had tried to exploit the same URL in at least one other organization that Datadog monitors.

Given that attackers frequently operate not by targeting specific applications but by broadly scanning the internet for easily exploitable vulnerabilities, keeping library dependencies updated with the latest patches is critical, as any known vulnerability represents a serious risk. In addition, due to the high prevalence of known-exploitable RCE and other vulnerabilities in Java libraries—and the higher likelihood that these vulnerabilities remain unpatched—teams should pay particular attention to remediating vulnerabilities in Java applications.

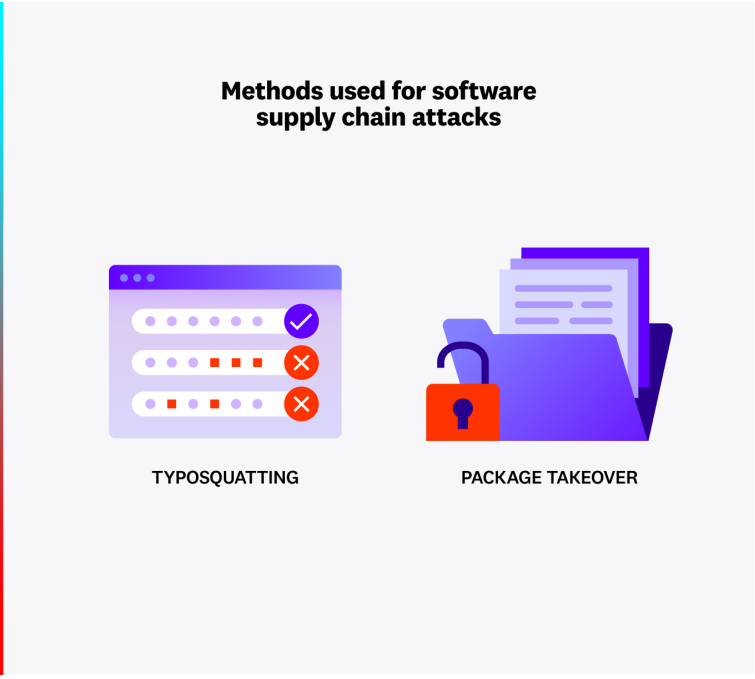
FACT 2

Attackers continue to target the software supply chain

In addition to searching web applications for known vulnerabilities, attackers also commonly attempt to trick developers into downloading and deploying malicious packages from open source libraries. Throughout 2024, Datadog identified thousands of malicious PyPI and npm libraries in the wild. Some of these packages were malicious by nature and attempted to mimic a legitimate package (for instance, `passports-js` mimicking the legitimate `passport` library), a technique known as typosquatting. Others were active takeovers of popular, legitimate dependencies (such as Ultralytics, Solana web3.js, and lottie-player). These techniques are used both by state-sponsored actors and basic cybercriminals.

BOTH STATE-SPONSORED AND BASIC CYBER CRIMINALS ARE ATTACKING THE SOFTWARE SUPPLY CHAIN

Source: Datadog



Attackers exploit the fact that vetting third-party dependencies is challenging for developers, especially if the package's metadata and functionality appear to be legitimate.

To mitigate the growing prevalence of malicious packages in open source ecosystems, we hope package manager attestations—such as those now available for npm and PyPI—and increased maintainer account security will become more widespread and decrease the risk that legitimate packages will be taken over by bad actors. In addition, open source tools such as GuardDog and Supply-Chain Firewall can help users identify malicious code in software packages before they get installed.

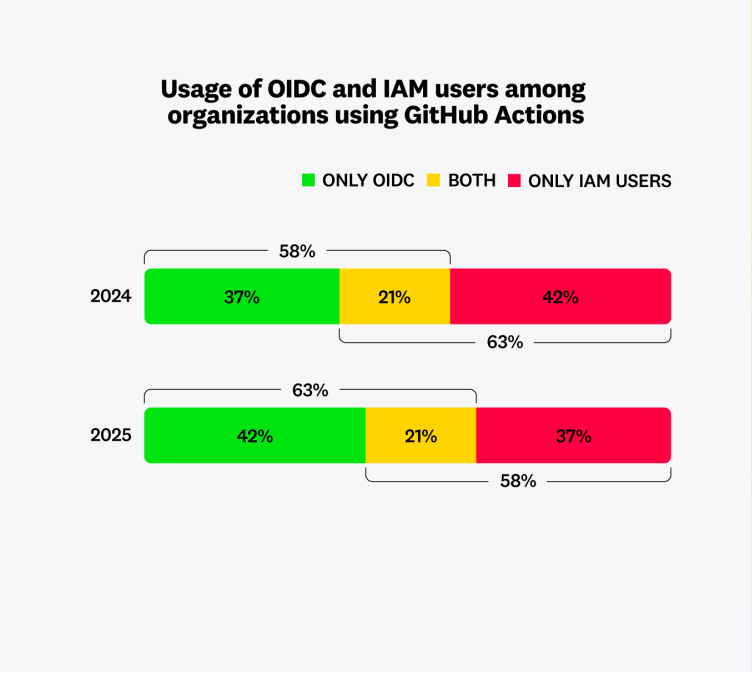
FACT 3

Usage of long-lived credentials in CI/CD pipelines is still too high, but slowly decreasing

In addition to vulnerable code and malicious open source packages, the use of long-lived credentials that don't expire is a common cause for cloud data breaches, and the practice is now widely frowned upon. Using long-lived credentials in CI/CD pipelines can be particularly risky, because their permissions are often highly privileged.

One scenario where long-lived credentials often make their way into CI/CD is when organizations use GitHub Actions to deploy applications to AWS, as these Actions can be configured with either short-lived credentials through OpenID Connect (OIDC) or AWS IAM user credentials, a type of long-term access key. We found that **37 percent of customers who send CloudTrail logs to Datadog are using IAM users in their GitHub Actions**. We strongly recommend not relying on IAM users in this scenario but instead using OIDC.

While 63 percent of organizations using AWS and GitHub Actions leverage OIDC (up from 58 percent in 2024), 58 percent of organizations also leverage long-lived credentials from IAM users (down from 63 percent in 2024).



DESPITE IMPROVEMENT, MOST AWS ORGANIZATIONS STILL USE LONG-LIVED CREDENTIALS IN GITHUB ACTIONS PIPELINES

Source: Datadog

This shows that long-lived credentials, although a well-understood source of risk, remain a major source of technical debt, as they are typically challenging to migrate to more modern identity solutions.

FACT 4

Only a fraction of critical vulnerabilities are truly worth prioritizing

The high prevalence of library vulnerabilities, malicious open source packages, identity misconfigurations, and other issues creates a noisy environment for defenders. In this context, security teams need to understand how to prioritize issues and determine which to remediate first. These teams often make decisions based on the severity of a vulnerability—typically measured by its CVSS base score, which is published along with the metadata for a known vulnerability. Vulnerabilities considered high-severity by CVSS score are only increasing: We found that the average service has **13.5 vulnerabilities with a high or critical CVSS severity score**, up from 11.9 in 2024.

However, in order to truly gauge a vulnerability's severity, it's essential to have the appropriate runtime context—for example, whether the vulnerability is running in a production environment, or if the application in which the vulnerability is found is exposed to the internet. CVSS does not take these factors into account, which leads to excessive noise for defenders.

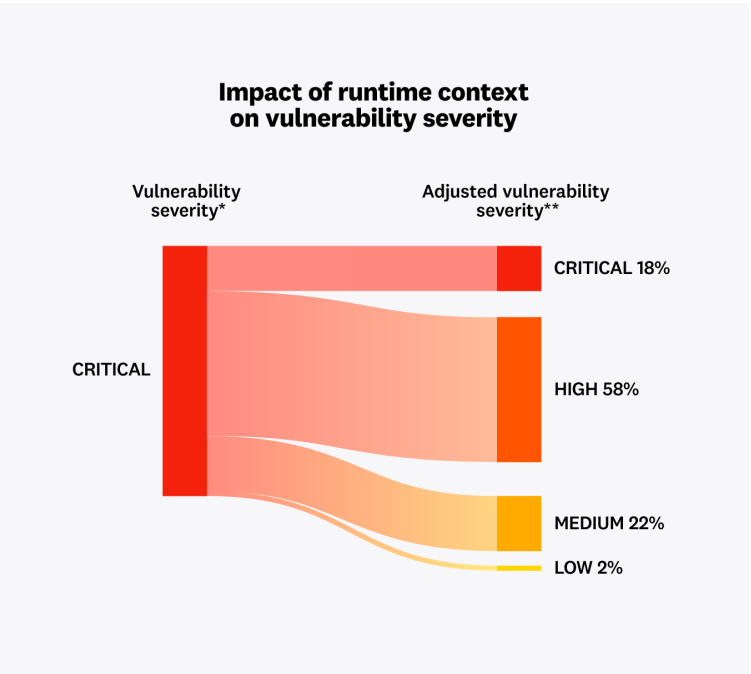
To reduce this noise, we developed a prioritization algorithm that factors in runtime context. After applying this algorithm, we found the average number of high or critical vulnerabilities per application drops from **12.2 to 7.5**.

The reduction in noise is even more significant when looking only at critical vulnerabilities—i.e., those that require the most urgent prioritization. Just **18 percent of vulnerabilities with a critical CVSS score—less than one in five—are still considered critical** after applying our algorithm.

RUNTIME CONTEXT SHOWS THAT LESS THAN 1 IN 5 CRITICAL VULNERABILITIES ARE WORTH PRIORITIZING

*From CVSS
**From Datadog App and API Protection

Source: Datadog



This dramatic reduction in severe vulnerabilities represents hours saved each week, month, and quarter—and if defenders spend less time triaging issues, they can reduce their organizations' attack surface all the faster. Focusing on easily exploitable vulnerabilities that are running in production environments for publicly exposed applications will yield the greatest real-world improvements in security posture.

"Without context, severity is just noise. True security comes not from patching everything, but from knowing what actually matters."

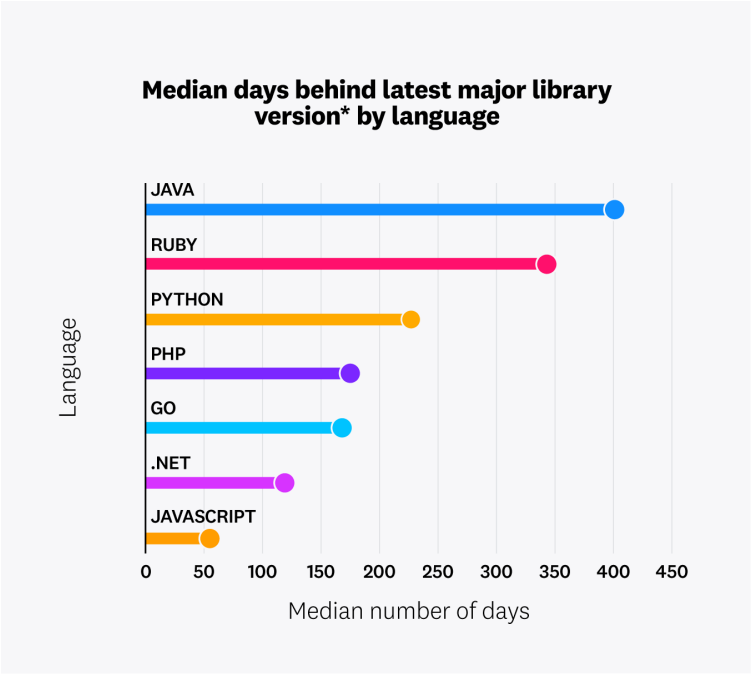
Jean Burellier
Principal Software Engineer at Sanofi

FACT 5

Keeping libraries up to date is a major challenge for developers

In our research, we found that the median dependency is **215 days behind its latest major version**. This reinforces the idea that engineers across DevSecOps functions are stretched and need to reduce noise, as teams struggle to keep library dependencies in their code up to date. Out-of-date libraries can increase the likelihood that a dependency contains unpatched, exploitable vulnerabilities.

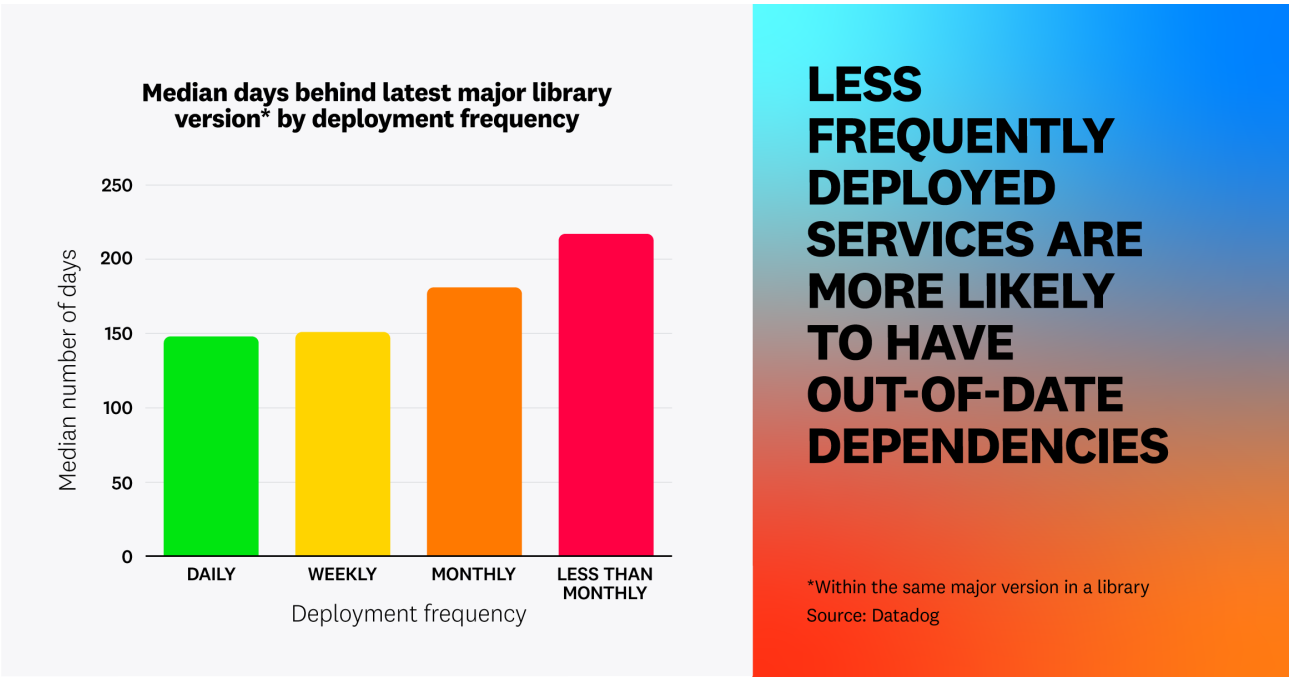
The trend that dependencies tend to be out of date is true for all ecosystems, and stronger for some. For example, while the median JVM dependency is behind by 401 days, the median for Go is 168 days.



ACROSS ALL LANGUAGES, DEPENDENCIES ARE MONTHS BEHIND THEIR LATEST MAJOR UPDATE

*Within the same major version in a library
Source: Datadog

Among all services, those that are less frequently deployed are more likely to be using out-of-date libraries. We found that **dependencies in services that are deployed less than once per month are 47 percent more outdated than those deployed daily**, with a median of 217 days versus 148 days behind their latest version update.



To complicate matters further, **one in two services use libraries that are not actively maintained**. Because updates are no longer being released at all for these libraries, any newly discovered vulnerabilities will remain unpatched.

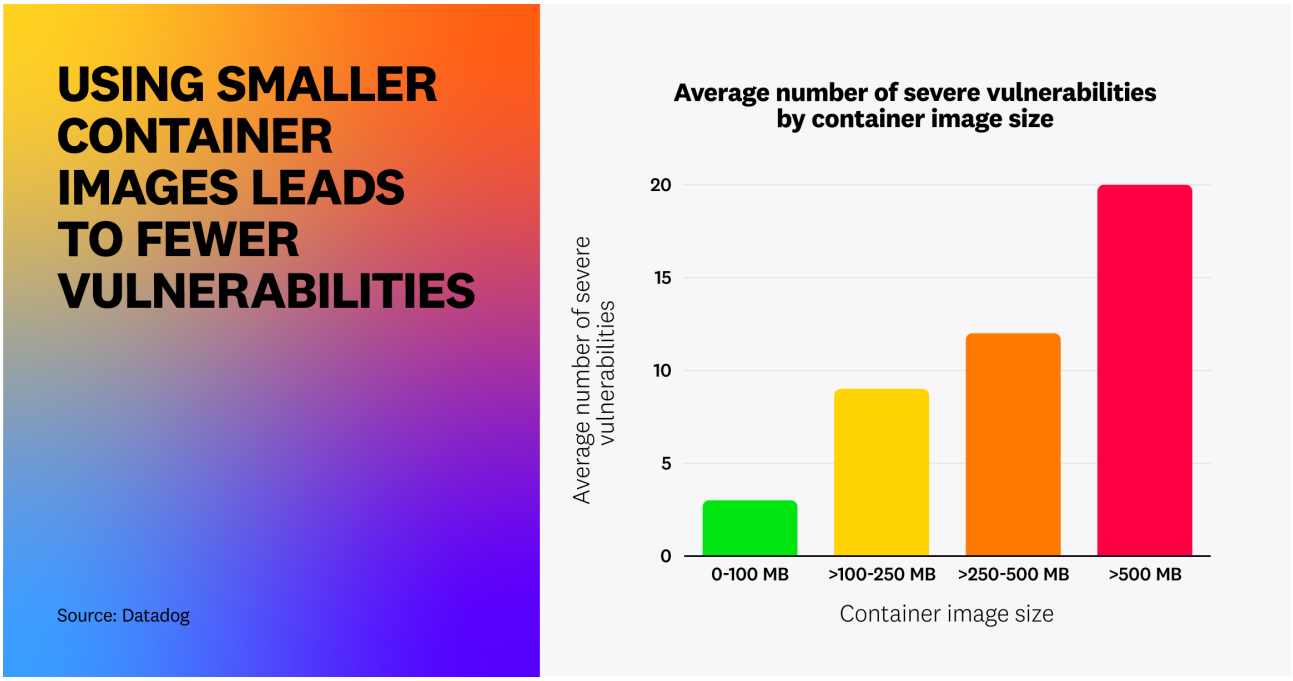
To lower exposure to vulnerabilities, teams should identify and update outdated libraries on a consistent basis. This holds especially true for unmaintained libraries, which should be treated like an end-of-life operating system and immediately updated to an actively maintained version. In addition, developers should deploy their services frequently. Not only is doing so an operational best practice, but because each fresh deployment gives teams a chance to upgrade their dependencies, it also simplifies the task of maintaining a strong security posture.

FACT 6

Minimal container images improve security posture

In recent years, engineering teams have moved toward adopting minimal container images in order to improve the efficiency and cost-effectiveness of their systems. This trend began with minimal distributions such as Alpine Linux and has evolved into even smaller distro-less and from-scratch images, all of which drastically reduce the size of a container image. For instance, the default Docker image for `python:3.11` weighs over 1 GB, while the minimal version `python:3.11-alpine` weighs 58 MB, reducing its size by 95 percent.

This "less is more" approach has benefits for security as well. By analyzing thousands of container images, we found that **on average, an image of less than 100 MB has three severe—i.e., with a high or critical CVSS score—vulnerabilities (median 0), while a container image of greater than 500 MB has on average 20 such vulnerabilities (median 7)**.



In addition to containing fewer vulnerabilities in the first place, minimal container images also make it harder for attackers to "live off the land" by using system utilities like `curl` or `wget` to perform exploits (which is a very common behavior), since lightweight images typically don't include this type of tooling. This reinforces the idea that teams should adopt lightweight container images wherever possible, both as an operational best practice and as a security measure.

“A smaller image means less to patch—and less to exploit. The more components you include, the larger the attack surface becomes. A distroless approach strips away all unnecessary packages and utilities, leaving only exactly what is needed for intended use; this minimizes vulnerabilities and ensures that attackers have far fewer footholds to work with. In our experience, minimal, hardened images not only cut down on CVEs but also help fast track compliance.”

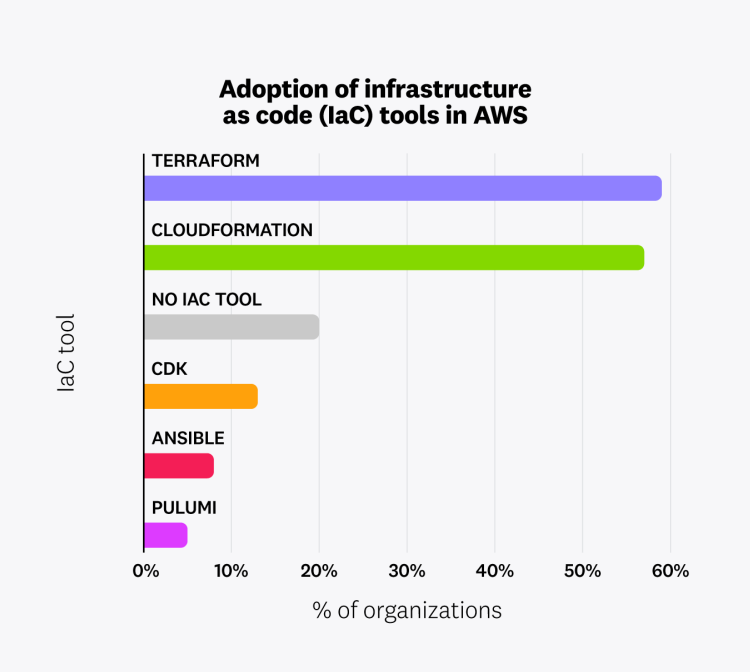
Amber Bennoui
Principal Product Manager, Product Security, Chainguard

FACT 7

In AWS, infrastructure-as-code usage is high, but many teams still use ClickOps as well

Infrastructure as code (IaC) has become the dominant practice in cloud environments because it tends to lead to better operational outcomes, such as improved version control and traceability. In addition, IaC is a key component of security best practices, as it helps enforce peer review and makes infrastructure easier to scan for misconfigurations.

IaC adoption continues to grow more widespread. In AWS, 59 percent of organizations are using Terraform, and 57 percent use CloudFormation (slightly up from 2024). In all, **80 percent of organizations use at least one IaC tool**.



IN AWS, TERRAFORM AND CLOUDFORMATION ARE BY FAR THE LEADING IAC TOOLS

Source: Datadog

At the same time, we also identified that at least **38 percent of organizations perform ClickOps deployments in production**—i.e., having engineers manually log in to their AWS accounts to provision infrastructure through the AWS Console. ClickOps is generally less secure than IaC, as it introduces more opportunities for human error and results in a wider group of engineers having privileged access to the cloud environment.

Our research also shows that **organizations using IaC are not exempt from ClickOps**, as some are using both methods. This demonstrates that IaC, while an essential operational and security practice, is not enough—people are still accessing production environments manually. It remains critical to enforce IaC, limit privileged access, and continually scan cloud resources for vulnerabilities.

Key learnings from the 2025 State of DevSecOps study

In this section, we outline several best practices that organizations should implement based on these findings, including to:

- Prioritize vulnerabilities with runtime context
- Deploy guardrails within your software supply chain
- Deploy frequently to stay current on patches
- Adopt minimal container images
- Expand IaC usage and rein in ClickOps

In addition, we explain how you can use Datadog Code Security, Cloud Security, Cloud SIEM, Workload Protection, and other products within the Datadog platform to improve your security posture and defend against threats.

Prioritize vulnerabilities with runtime context

Most security teams have a finite set of resources they can leverage to remediate security vulnerabilities, which means that prioritization is key. Luckily, not all vulnerabilities are equally urgent. CVSS base scores help with prioritization, but these scores do not account for environmental attributes, such as public accessibility or whether or not the vulnerability has a known exploit available. Unfortunately, trying to manually apply this type of context to thousands of vulnerabilities is a time- and labor-intensive task that is simply not possible for most teams.

However, by mapping environmental information about your cloud assets to the vulnerabilities associated with each resource, you can use runtime context to effectively prioritize the vulnerabilities that are the most likely to get exploited. Here are some key characteristics to consider when adjusting severity ratings:

- **Public accessibility:** Vulnerabilities present in internet-facing systems are more likely to be identified and exploited. Vulnerabilities in systems that are not public-facing are certainly relevant, as they can be exploited once an attacker has already gained initial access to your environment. But when it comes to prioritization, organizations should focus first on vulnerabilities that affect public-facing systems.
- **Environment:** A vulnerability exploited in a production environment is much more likely to lead to service disruptions, sensitive data breaches, and further compromise of additional production services, as production systems contain the most sensitive data and live user sessions. A vulnerability in a development or staging environment is generally less critical than one in production.

- **Active attack status:** Prioritize vulnerabilities in any environments that you know are currently being targeted by malicious activity, including automated scanners, exploit attempts, or other indicators of compromise.
- **Highly privileged identity:** Vulnerabilities present on systems or endpoints that have highly privileged identities associated with them pose a higher risk because they make post-exploitation tasks significantly easier for the attacker. Highly privileged identities, such as admin users, grant access to more sensitive data and critical system functions. If these assets are compromised, attackers can escalate privileges and gain control over significant portions of the infrastructure.
- **Known exploit proof of concept:** Vulnerabilities with known exploits are significantly more dangerous, as attackers have readily available tools and methods to exploit them. These vulnerabilities are actively targeted and can lead to rapid compromise, even if the underlying system is not directly internet-facing. Prioritizing these vulnerabilities is crucial due to their high likelihood of exploitation and the potential for widespread impact.

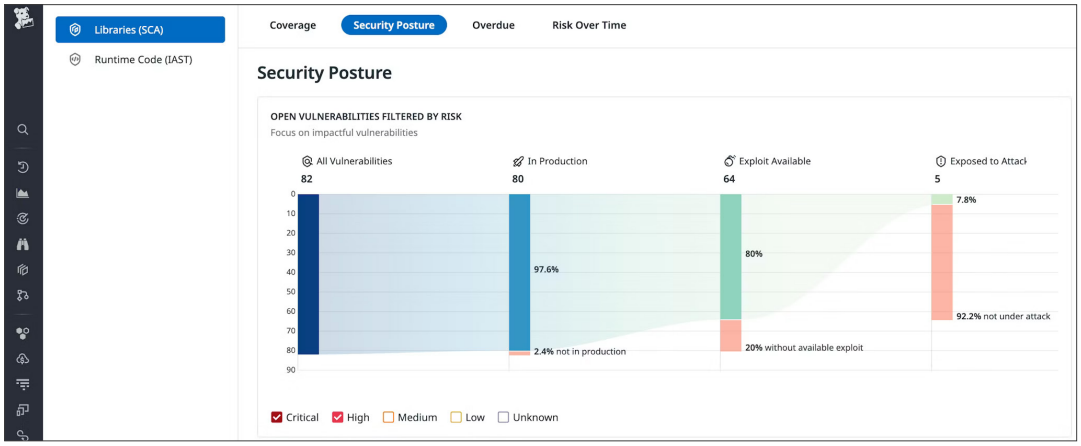
By taking these characteristics into account, you can effectively prioritize vulnerabilities and allocate resources to address the most critical risks first. This approach ensures that your security efforts are focused on protecting your most valuable assets and mitigating the most significant threats.

Use Datadog to prioritize vulnerabilities and threats using runtime context

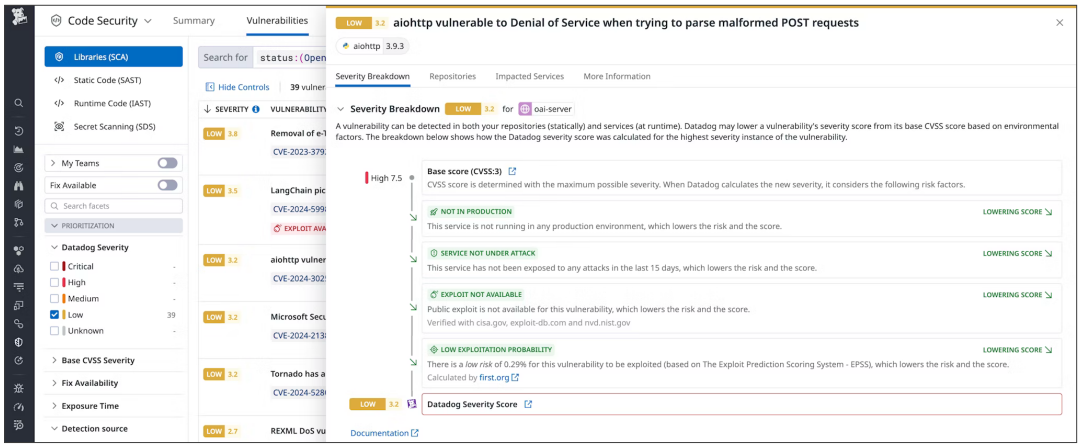
Datadog's security portfolio enriches and correlates security findings with runtime insights from your environment. This helps you understand the actual impact of vulnerabilities on critical workloads and make smarter remediation decisions. Datadog applies runtime context in two key areas to reduce alert noise: Security Inbox and Datadog Severity Scoring.

Datadog Severity Scoring layers in critical environment context like exploitability, accessibility, and likelihood of attack to base CVSS scores. Datadog may increase or decrease this score based on context from your specific environment. This helps teams prioritize what to fix first.

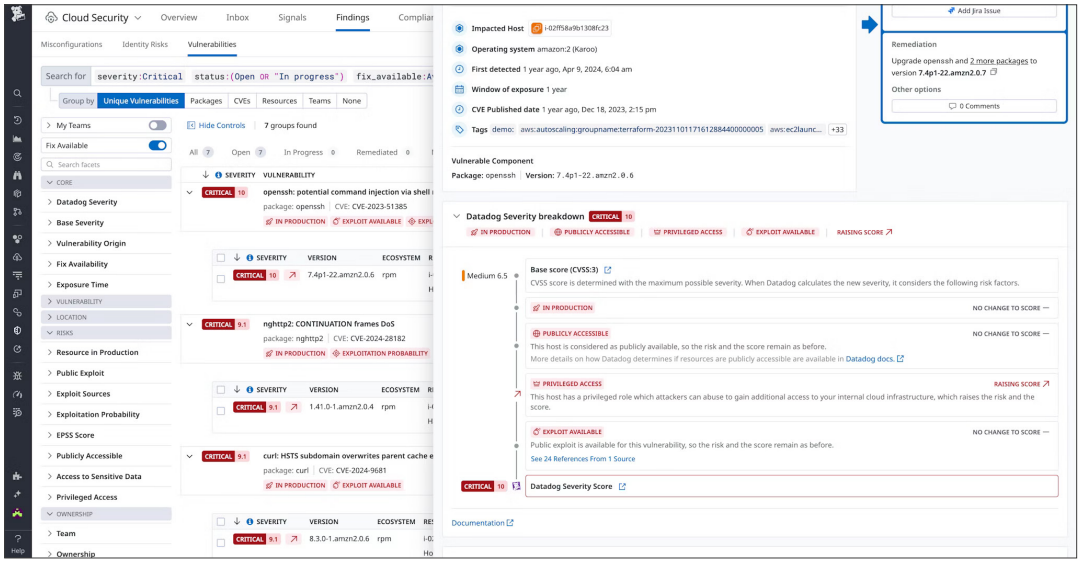
You can see this in action in the security posture funnel within Datadog Code Security. In the example below, looking only at vulnerabilities that are running in production, have an exploit available, and were exposed to attacks has reduced the number of critical vulnerabilities by more than 92 percent.



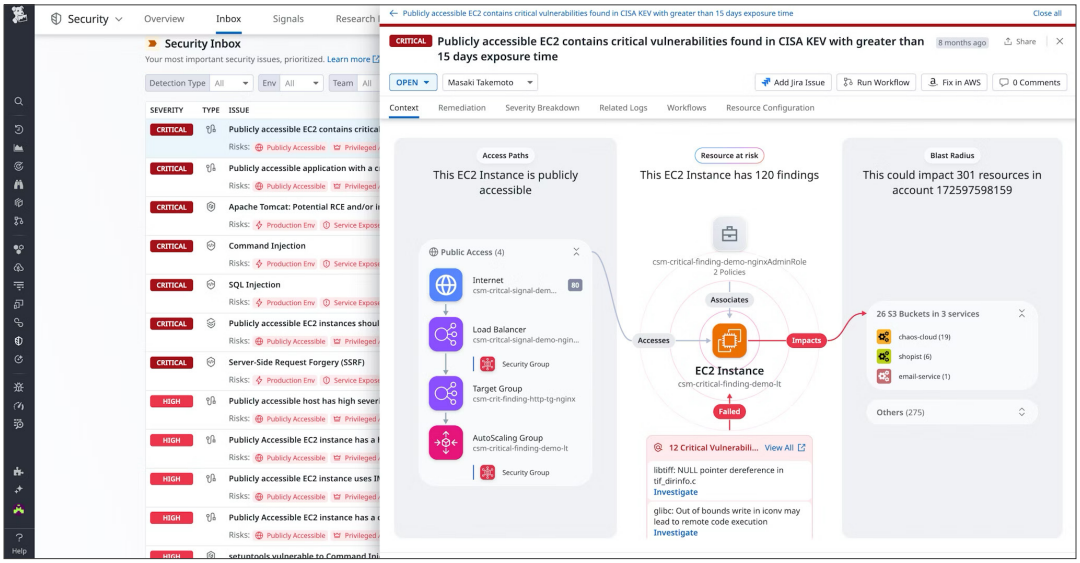
Here's another example of Datadog Severity Scoring. The image below shows a vulnerability within an aiohttp library that had a base CVSS score of 7.5. However, Datadog had the environmental context to see the service was not in production, not under attack, had no exploit available, and was unlikely to be exploited based on the Exploit Prediction Scoring System (EPSS). Based on this context, Datadog reclassified the vulnerability from high severity to low.



Here's an example of the same prioritization happening in Datadog Cloud Security. This time, we're looking at a vulnerability on an EC2 instance that had a base CVSS score of 6.5. Normally, this wouldn't raise any alarms or be considered an urgent remediation. However, Datadog can see that this EC2 instance has a privileged role, which an attacker could abuse to gain additional access to cloud infrastructure. This raises the score significantly and informs security teams that this vulnerability could be more impactful than it initially seemed.



Datadog also prioritizes findings using Security Inbox, which provides an actionable list of your most critical findings and their relationship to each other. Security Inbox automatically correlates insights from across Datadog, including vulnerabilities, signals, misconfigurations, and identity risks. This allows you to see the most critical combinations of risks that could create viable attack paths for a threat actor to leverage.



Deploy guardrails within your software supply chain

Attackers continually evolve techniques to trick developers into installing malicious libraries by targeting the ecosystems that support modern software development, such as npm and PyPI. Successful execution of these attacks can occur anywhere, from developer laptops to your organization's CI/CD pipelines.

Mitigating this risk requires a multi-pronged approach. You can take actions to prevent or detect the execution of malicious packages within your trusted environments, and you can also protect software repositories to harden your ecosystem against attacks.

In terms of actions you can take, you'll want to scan software packages as they are used to determine if they are malicious. For example, a solution like Datadog's Supply-Chain Firewall can use known databases, such as Datadog Security Research's public malicious packages dataset and OSV.dev, and prevent the execution of any packages marked as malicious within those databases. Additionally, you can use GuardDog to identify malicious packages holistically and even extend its capabilities with additional signatures to detect malicious patterns in packages.

As security practitioners, we must always operate with the assumption that prevention eventually fails, so it is also prudent to also deploy traditional endpoint detection and response (EDR) tools to detect malicious activity on developer workstations. Additionally, you'll want to ensure you have visibility in your CI/CD pipelines and cloud workloads so you can detect malicious activity there as well.

The other approach to tackling these threats is to apply pressure on the owners of software repositories and marketplaces to deploy the types of controls that prevent attacks at the source. For example, an ecosystem that provides package maintainer attestations or providence statements allows users to verify these attestations at time of use, which helps prevent the execution of any software that has not been signed by the author.

Secure your pipeline by eliminating long-lived credentials

Leaks of long-lived cloud credentials are one of the most common causes for cloud breaches—because they never expire, any leak of these types of credentials in configuration files, source code, or another source could provide a viable entry point for attackers. That's why it's critical to use short-lived cloud credentials through OpenID Connect (OIDC) or a similar protocol to authenticate both workloads and human users to cloud resources.

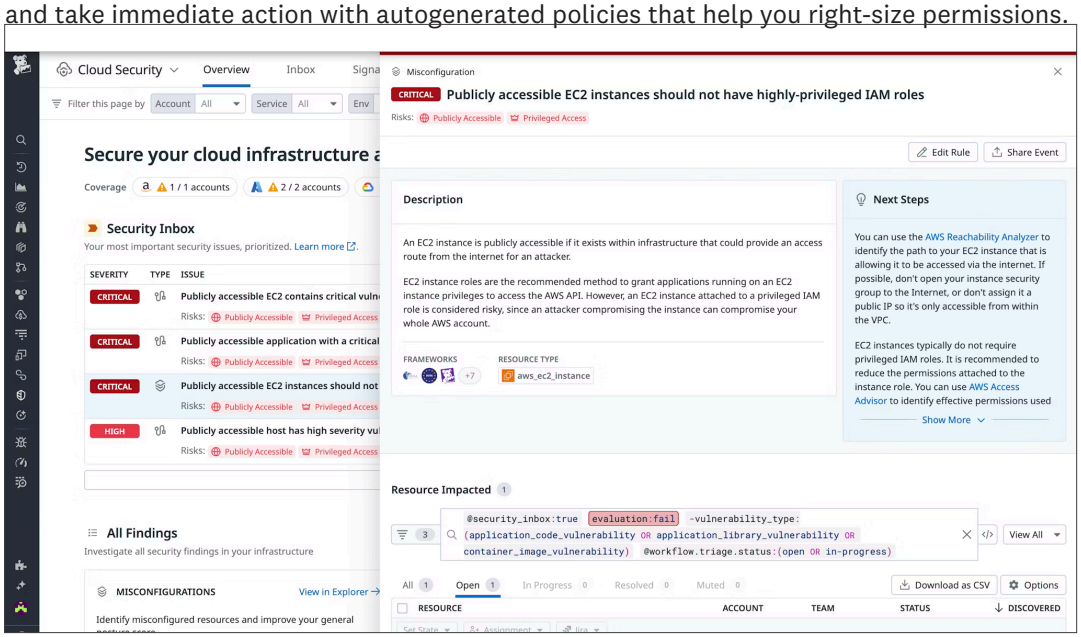
It's particularly important to use short-lived credentials in CI/CD pipelines. These workloads often run on external infrastructure and need to authenticate to cloud environments to deploy resources, typically through infrastructure-as-code (IaC) or some sort of automation. Here's a look at how you can use short-lived cloud credentials in your CI/CD provider through the OIDC protocol:

- The CI/CD provider injects into the running pipeline a signed JSON Web Token (JWT) that provides a verifiable identity to the pipeline.
- The CI/CD pipeline uses this JWT to request cloud credentials. How this is done depends on the cloud provider. On AWS, for instance, this is achieved using a call to AssumeRoleWithWebIdentity.
- If the cloud environment has been properly configured to trust and distribute credentials to this pipeline, it returns short-lived credentials that are valid for a few hours.

In addition, you'll want to monitor all of your systems for identity risks such as overprivileged users and unused credentials.

Use Datadog to monitor authentication events

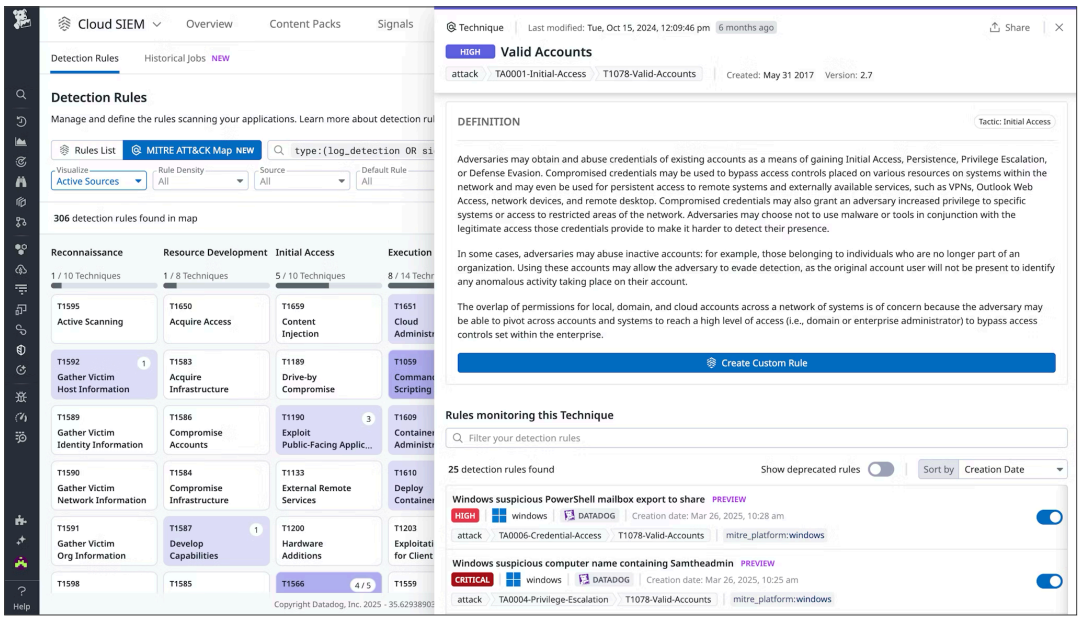
Datadog provides multiple ways to monitor your cloud environment for suspicious authentication activity or potential misconfigurations. For example, Cloud Security provides out-of-the-box rules to identify risky IAM users with long-lived, unused credentials. In addition, you can gain visibility into accessible resources with access insights. This allows you to see what entities a resource can access and who can access the resource. You can also get granular insights into identities that have direct or indirect access to a given resource and take immediate action with autogenerated policies that help you right-size permissions.



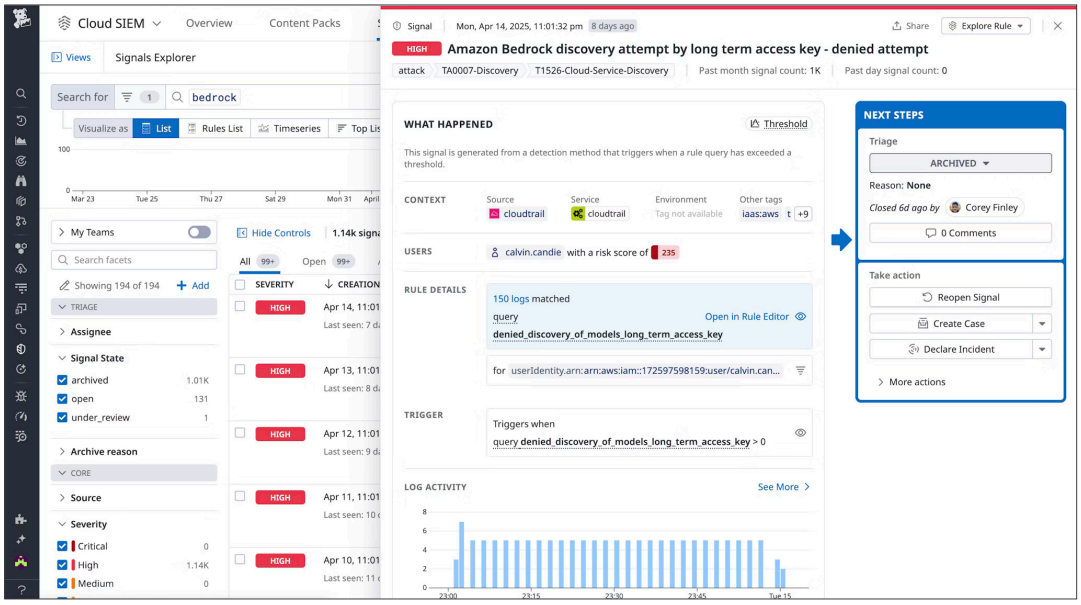
In addition, Datadog Cloud SIEM can help you identify compromised access keys through numerous out-of-box rules, including:

- Compromised AWS IAM user access key
- TruffleHog user agent observed in AWS, which identifies an attacker who successfully uses the popular TruffleHog tool to discover leaked access keys
- The AWS managed policy AWSCompromisedKeyQuarantineV2 has been attached, which identifies when AWS itself quarantines an IAM user whose access key was publicly leaked

You can also easily identify gaps to strengthen detection coverage with the MITRE ATT&CK Map and create custom Cloud SIEM rules using new value, anomaly detection, impossible travel, and other detection methods to identify unexpected activity.



For example, you can create a rule to alert when an attempt to access Amazon Bedrock is made using a long-term AWS access key.



Lastly, App and API Protection helps filter through the noise and find suspicious authentication patterns that may indicate credential stuffing and account takeover campaigns.

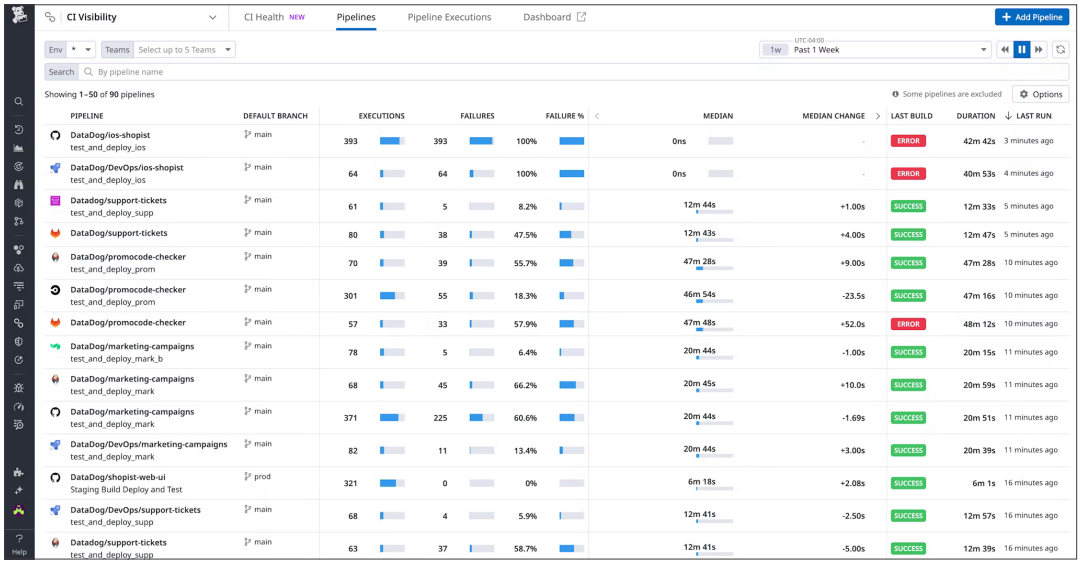
Deploy frequently to stay current on patches

Services that are deployed more frequently contain fewer out-of-date dependencies. One straightforward takeaway here could be that—even if your code has not changed—simply executing CI tasks that update dependencies, build, and test your software on a regular cadence can significantly improve the security of your services without much manual effort. That said, there is plenty of debate about the risks and rewards of automating dependency updates, so it's worth spending time to understand if this type of solution can work for you.

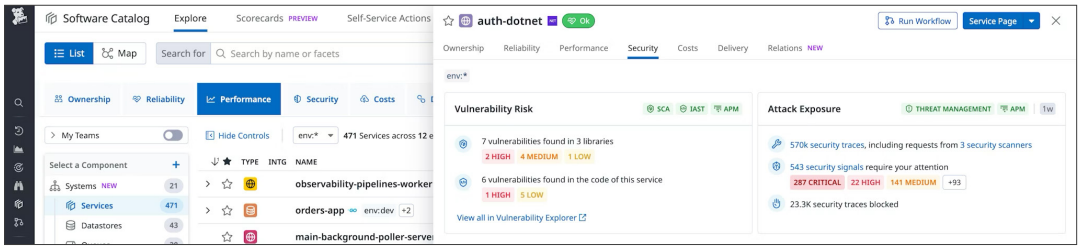
As impactful as automated dependency updates can be, there's still another area that needs to be addressed. We found that one in two services use libraries that are not actively maintained, which means they will no longer receive security updates, even if new vulnerabilities are identified. To tackle this problem, we recommend keeping track of all end-of-life libraries currently in use within your organization and starting with these end-of-life dependencies when prioritizing remediation efforts.

Monitor service health with Datadog

Datadog provides a range of capabilities that enable you to view the health of services so you can accelerate deployments and upgrade dependencies more frequently. CI Pipeline Visibility supports development velocity by giving you a high-level view into the health of your CI/CD systems.



You can also use Software Catalog to attain a centralized, real-time view of your service ecosystem, so your teams can track ownership, monitor performance, and enforce security and compliance standards all in one place. From here, you can also see the security posture of each service, including active threats like SQL injection and vulnerabilities in application code or libraries.



Adopt minimal container images

Most modern applications run in cloud-native containerized systems. Whether they run on top of Kubernetes or provider-specific services such as Amazon ECS or Azure Container Instances, these applications are packaged within a container image. When building such applications, it's important to attempt to build small, minimal container images: They are faster to deploy, remove complexity, and drastically reduce the number of OS-level vulnerabilities.

There are a number of possible approaches to building minimal container images. The first might be to use profiling software like slim to analyze images and remove unnecessary software from them. This kind of approach works well with existing images, where re-architecting from scratch might not be possible.

When building new images, it's important to use a minimal base image. If the application doesn't require a full Linux distribution to run, there are options like Docker's scratch image, which provides an effectively empty starting point for applications. Alternatively, Distroless provides a starting point with some more base Linux files, which some applications need in order to operate effectively.

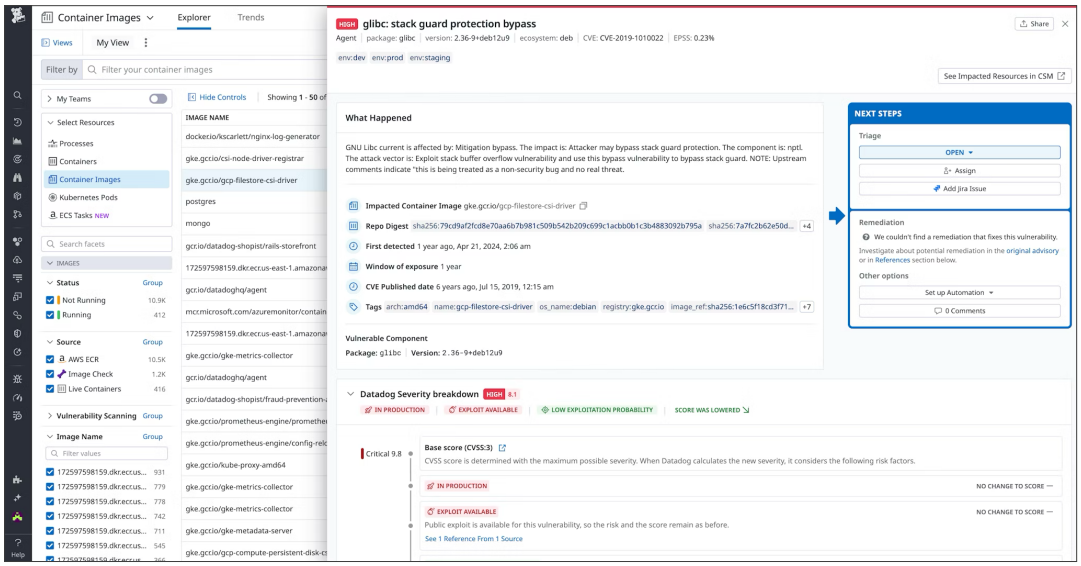
If your application requires more operating system support files, be careful to avoid installing unnecessary packages, such as development dependencies. You can use techniques like multi-stage builds to keep the final container image as small as possible.

Monitor the security of your container images with Datadog

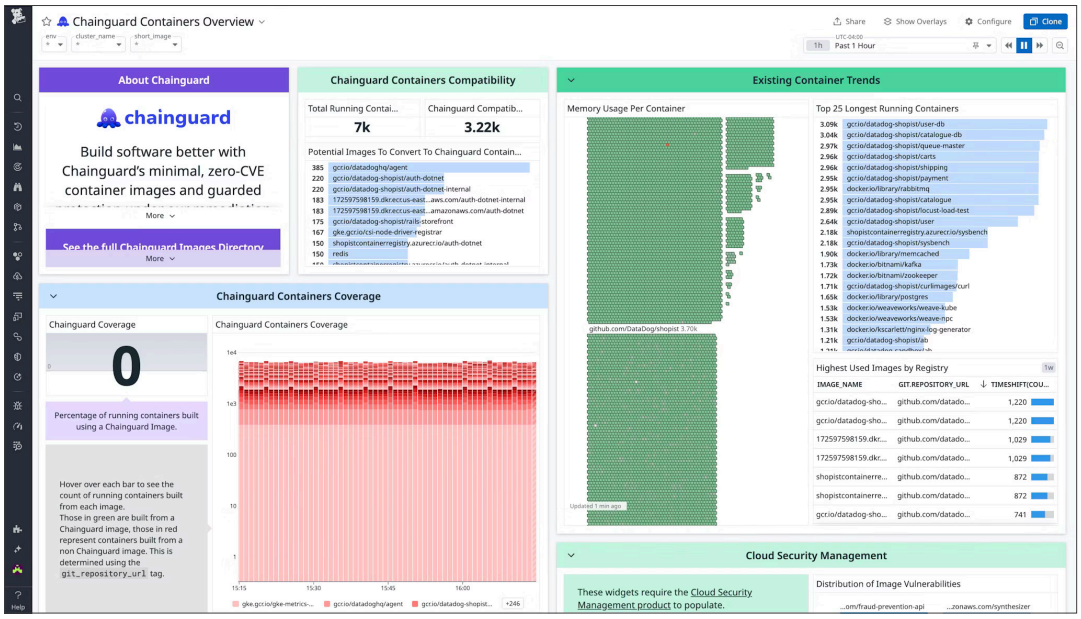
From development through production, Datadog offers deep visibility into the security posture of running container images. With security insights embedded in Infrastructure Monitoring views, engineers and resource owners are better equipped to secure containers as part of their standard optimization and deployment workflows.

The Container Images view provides rich information about the size of container images, their age, and any detected vulnerabilities. By clicking into specific images, you can immediately see all containers where the image is deployed, alongside their associated status and CPU usage. This ensures you have full context on where each image is running and how it's performing, so you can make faster, more informed remediation decisions.

When investigating vulnerabilities on a container, users can see detailed insights like the repo digest, resource tags, and a timeline of when vulnerabilities were introduced. They can also automate remediation workflows with built-in workflow automation tools. This allows engineers to proactively remediate vulnerabilities that could impact performance reliability without having to wait for the security team to come to them with a list of critical remediations.



Additionally, Datadog's integration with Chainguard allows users to see widely adopted vulnerable container images and identify places where Chainguard container images could be used to lower that risk. This ensures that high-value container images are minimal and secured before they go into production.



Expand IaC usage and rein in ClickOps

The concept of Zero Touch production environments was first introduced in 2019 by Michał Czapiński and Rainer Wolafka at a USENIX conference. At a high level, these environments rely heavily on reliable automation as opposed to manual, error-prone actions performed by humans. Zero Touch production also promotes the concept of the least privilege: When an automation can perform a workflow (such as deploying a cloud workload), human operators don't need direct access to the production environment and can instead rely on safe, indirect, and user-friendly interfaces.

The first step toward implementing Zero Touch (or "low-touch") production environments is to automate the deployment process, typically using IaC technologies. Tools like Terraformer can also help convert an existing environment into its code representation.

Once automation is in place, operators typically need only minimal permissions (such as view only) to the production environment, since provisioning new infrastructure now happens through automation. It's helpful to plan "break-glass" roles ahead of time—these are roles that can be activated with approvals and audits when a specific situation, such as an incident, requires an operator to use privileged permissions in a production environment.

Track manual cloud actions with Datadog

It's important to be notified when there are manually initiated changes in your cloud environment. You can use the following query in Datadog Cloud SIEM to identify and alert on manual actions performed from the AWS Console (adjusted from Arkadiy Tetelman's methodology):

In addition, Datadog provides advanced real-time monitoring and security capabilities across application, API, and workload layers. At the workload level, Datadog Workload

```
1. source:cloudtrail @http.useragent:("console.amazonaws.com" OR
   "Coral/Jakarta" OR "Coral/Netty4" OR "AWS CloudWatch Console" OR
   S3Console/* OR \[S3Console* OR Mozilla/* OR onsole.*.amazonaws.com
   OR aws-internal*AWSLambdaConsole/*) -@evt.name:(Get* OR Describe*
   OR List* OR Head* OR DownloadDBLogFilePortion OR TestSchedule-
   Expression OR TestEventPattern OR LookupEvents OR listDnssec OR
   Decrypt OR REST.GET.OBJECT_LOCK_CONFIGURATION OR ConsoleLogin) -@
   userIdentity.invokedBy:"AWS Internal"
```

Protection utilizes eBPF-based, in-kernel instrumentation to provide deep visibility into system-level activity. This includes monitoring sensitive file access, network connections, and process execution with minimal performance overhead. These activity signals are correlated to detect suspicious behaviors and unauthorized actions, supporting threat detection, drift prevention, and incident response at runtime.

Secure your SDLC with Datadog

To ensure your software development practices support high release velocity while keeping applications and systems protected against threats, you need end-to-end visibility into vulnerabilities in your code, misconfigurations across your cloud resources, and the context to prioritize risks to remediate. Datadog surfaces these risks in the same platform where you monitor system health and performance, breaking down silos between teams so you can accelerate your adoption of DevSecOps best practices and improve security outcomes.

Check out our documentation to get started. If you're not a customer, you can get started today with a 14-day free trial.

Methodology

Fact 1

For this fact, we analyzed vulnerabilities in third-party libraries of applications across various languages and runtimes (Java, .NET, PHP, Python, Ruby, Javascript, and Go) and that use Datadog Code Security's Software Composition Analysis feature. We sourced known exploited vulnerabilities from the CISA KEV catalog, which we extracted on April 9, 2025.

For time to patch, we calculated the time it took for organizations to patch their vulnerabilities in each ecosystem.

We define "untargeted" as an instance in which the same attacker (identified by source IP) sent the same HTTP request to at least two Datadog customer orgs.

Fact 2

For this fact, we relied on the research and efforts of Datadog's security research team. Using tools like GuardDog, they identified and classified malicious packages across multiple software ecosystems since the publication of the 2024 version of this report.

Fact 3

To identify organizations that use GitHub Actions with OIDC authentication, we queried AWS CloudTrail logs using the following Datadog logs query:
`@eventName:AssumeRoleWithWebIdentity @userIdentity.identityProvider:*token.actions.githubusercontent.com* -status:error`

To identify organizations that use GitHub Actions with IAM users, we queried AWS CloudTrail logs using the following Datadog logs query:
`@userIdentity.accessKeyId:AKIA* @userIdentity.type:IAMUser -status:error`. We also filtered the results on source IPs known to be used by GitHub Actions, as determined by GitHub's API endpoint.

Fact 4

We analyzed vulnerabilities in third-party libraries called by applications that use Datadog Code Security's Software Composition Analysis feature.

We considered vulnerabilities with a "critical" CVSSv3 base score and, based on the context available, used the following methodology to compute the "temporal" and "environmental" CVSSv3 metrics:

- When the service was running in a non-production environment, we adjusted the "modified confidentiality, integrity, and availability impact" to "low."
- When the service was not publicly exposed on the internet and the exploit vector was "network," we set the "modified attack vector" to "local."
- When the EPSS score was below 1 percent, we set the "modified attack complexity" to "high."

– When a public exploit was available, we set the "exploit code maturity" to "proof of concept" or to "unproven" otherwise.

We then computed the adjusted score based on the CVSS v3.1 methodology and considered the ratio of vulnerabilities whose adjusted score was still "critical."

Fact 5

For this fact, we collected data on libraries that were active in March 2025. For each dependency, we calculated the number of days since the latest update to the current major version of a library was released. We then computed median lags across ecosystems (e.g., JVM, Go) and also compared them by deployment frequency.

We only looked at libraries that follow Semantic Versioning and excluded public libraries from our analysis.

For frequency of deployment, we obtained the data from the `datadog.service.time_between_deployments` metric, looking at deployments in the past six months. If a service was deployed 100 times or more in the last six months, we counted the service as being deployed daily, as there are 100 business days in six months. If a service was deployed 26 times or more in six months, we counted it as being deployed weekly. If the service was deployed six times or more, we counted it as being deployed monthly, and we categorized services deployed less than six times in six months as being deployed less than monthly.

Fact 6

We analyzed data from containers scanned through Datadog Cloud Security's Vulnerability Management feature and reviewed any identified OS-level vulnerabilities. This includes both publicly available images and images from private registries with at least one vulnerability. We excluded Datadog container images.

Fact 7

For this fact, we analyzed AWS CloudTrail Logs and determined which IaC technology was used based on the HTTP user agent.

Note: The data window for this fact is from the month of March 2025. If an organization did not use a known IaC technology during this period, we counted them as "not using IaC."

To determine how many organizations were using ClickOps, we analyzed AWS CloudTrail logs. Specifically, we defined that an organization is "performing manual cloud deployments through the AWS Console" if we found at least one of the events from the list below in all the AWS accounts they monitor with Datadog. Because we assume that every organization monitors at least one production account with Datadog, we're able to determine that organizations that meet this criteria use ClickOps in their production environment.

`RunInstances`
`AuthorizeSecurityGroupIngress`
`CreateVpc`
`CreateCluster`
`CreateDBCluster`
`CreateDBInstance`
`CreateInstances`
`CreateKeyPair`
`RegisterTaskDefinition`

We then filtered to identify when these events were performed manually from the AWS Console using the methodology described by Arkadiy Tetelman.

Licensing

datadog.com