Beyond the Hijack:
A Guide to Proactively Securing your npm Dependencies with JFrog Curation

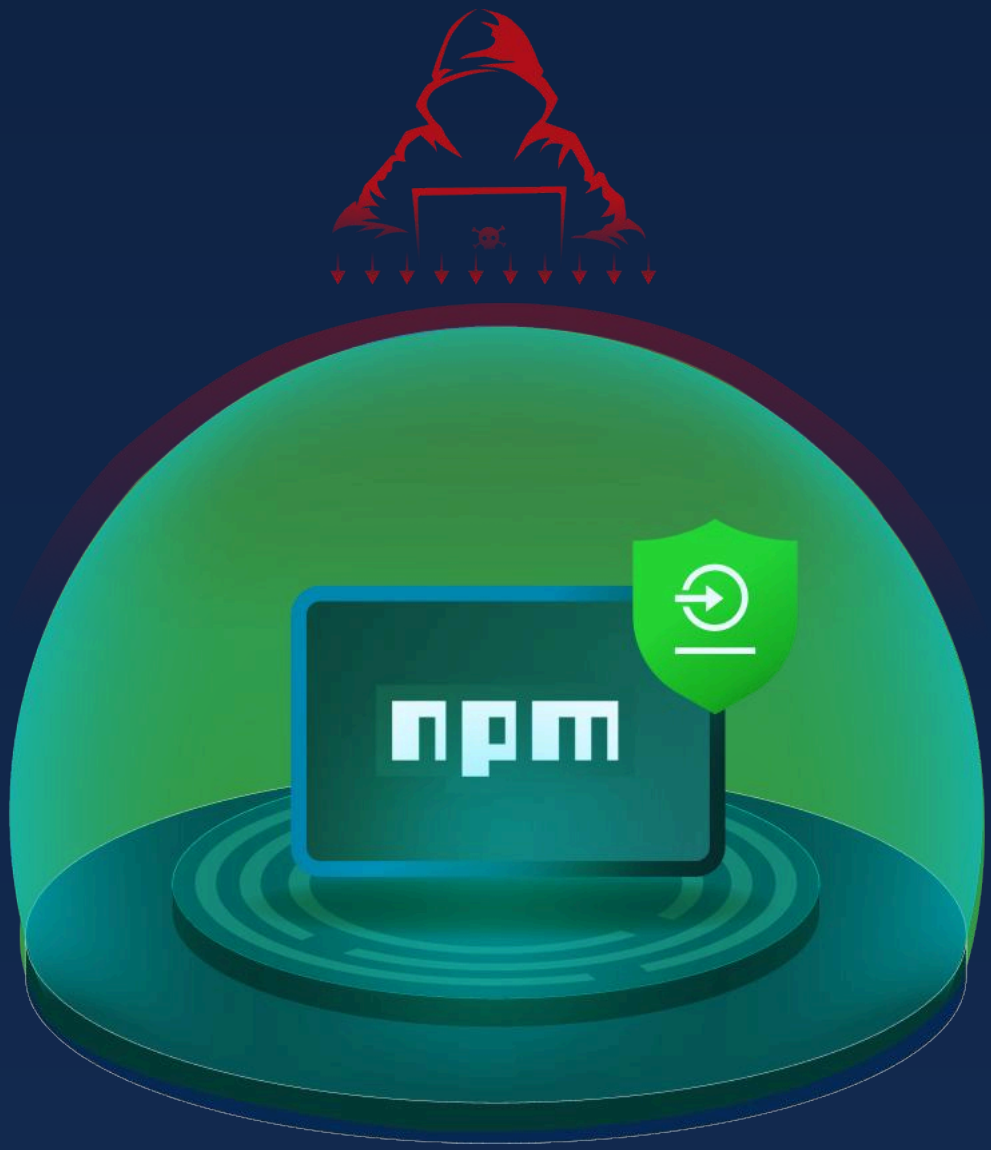

Table of Contents

INTRODUCTION

In September 2025, the developer community witnessed the largest npm supply chain attack in history. The result of two major attacks compromised over 200 popular packages with over 500 malicious versions, accounting for more than 2 billion weekly downloads. Attackers gained access by stealing a single developer's credentials, allowing them to upload malicious versions of these trusted packages for unsuspecting users to download. The severity of this attack and the number of developers affected, highlights a critical flaw in most DevSecOps programs, where application security has become a reactive, rather than proactive process.

Attackers are masters at exploiting the window of opportunity between the time a new open-source (OSS) package is made available and before its vulnerabilities or malicious nature is discovered. To truly secure your software supply chain, a strategic shift is required. Some even suggest that application security must move **"lefter than left"** to which vulnerabilities post an actual threat and preemptively blocking ' risky' packages before they ever enter your development environment.

In the case of the recent npm attack, the threat wasn't just theoretical - it was real. Fortunately, **JFrog Curation customers with the relevant policies in place, were completely protected** as these malicious packages were blocked automatically and not allowed to enter the development ecosystem. In terms of adopting a shift-left approach, the pre-emptive exclusion of these packages really demonstrates the effectiveness of how proactive application security can work in a real-world scenario.

This guide provides a step-by-step playbook for implementing a proactive defense that can help protect your organization from current and future software supply chain attacks.

UNDERSTANDING THE ATTACKER'S ADVANTAGE: THE HIJACK LIFESPAN

In order to defeat an attacker, we must understand their strategy and how they think. The JFrog Security Research Team investigated this incident in particular, and researched the general lifespan of hijack attacks in general. Their findings confirm that attackers rely on both speed and the inherent trust developers place in familiar packages
In the npm hijacking attack,malicious code was injected to intercept and divert cryptocurrency transactions from unsuspecting users and systems.

They succeeded by adding their malicious code to otherwise legitimate well known third party packages. This tactic is effective because it also exploits a well known time gap. Developers, driven by the need for new features and urgent bug fixes, quickly adopt the latest version of familiar packages without checking for potential vulnerabilities.

At the same time, security agencies playing catch-up rely on public vulnerability databases, which are only updated after a threat is discovered and analyzed. This defensive posture guarantees you will always be a step behind. The only way to win is to eliminate the attacker's window of opportunity altogether.

THE PROACTIVE PLAYBOOK: PROTECTING YOUR SOFTWARE SUPPLY CHAIN WITH JFROG CURATION

JFrog Curation gives you automated control to ensure only approved open source packages are used by your developers. Here are the three steps taken by JFrog customers to set up this critical line of defense.

Step 1: Block the Primary Attack Vector with an "Immature Package" Policy

Guideline: JFrog's security research team studied hijacking attacks and found that most attacks are discovered within 48 hours, with more sophisticated attacks usually discovered within the first 14 days. This creates a 14-day window for the attacker's lifespan. By implementing an 'immature package' policy that blocks downloads of packages that are 14 days or less from release, JFrog significantly reduces the risk from this entire class of attacks.

Action: This requires implementing a policy that blocks packages that have been available for download for 14 days or less, which is a common window for exploits to be discovered

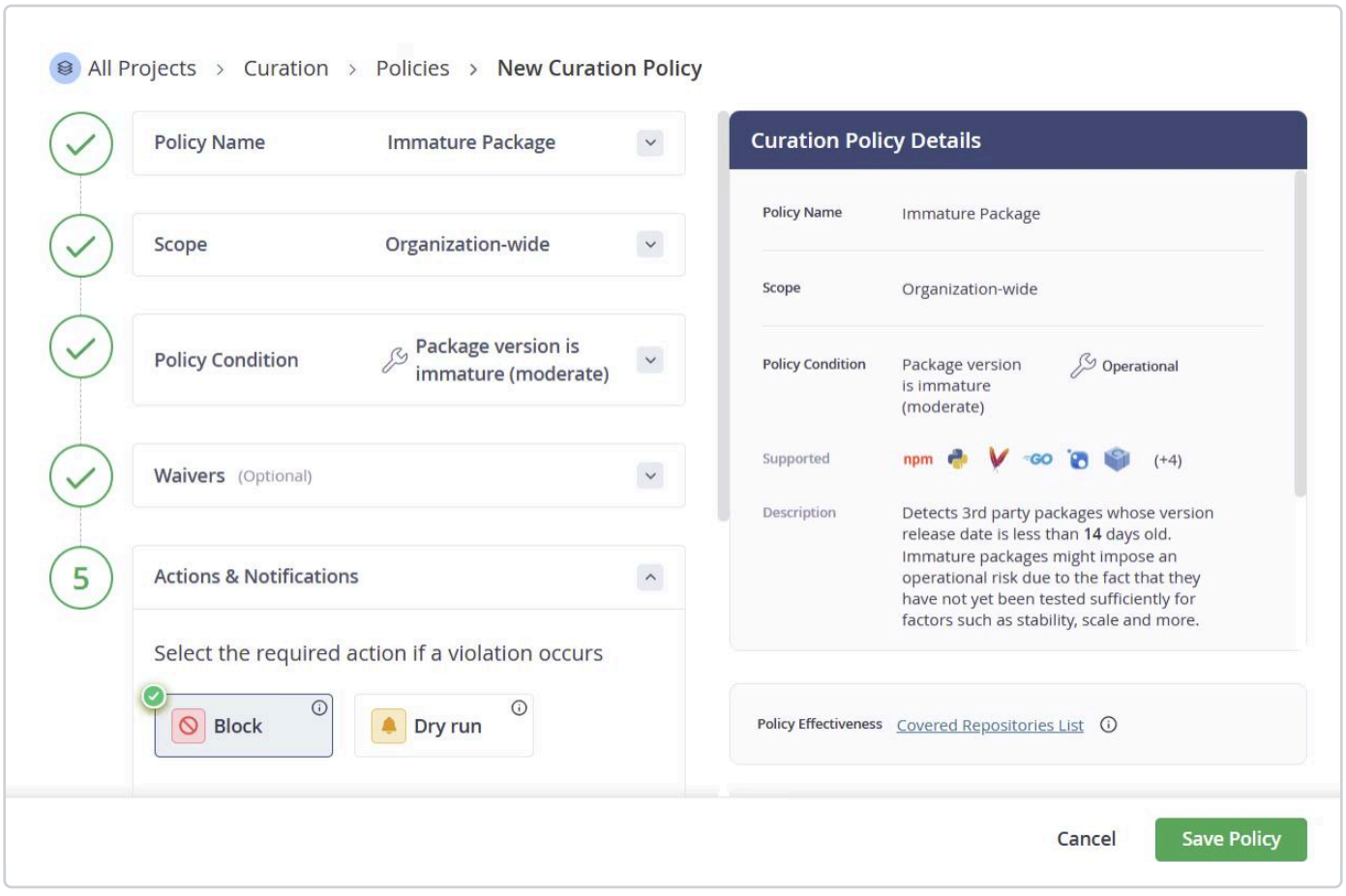

FIGURE 1: Creating detailed policies blocking the use of immature packages in JFrog Curation

- In the JFrog Platform, navigate to **All Projects > Curation > Policies > New Curation Policy.**
- Set the **Policy Name** to something descriptive, like "Immature Package".
- Define the **Scope** as "Organization-wide" for maximum protection.
- For the **Policy Condition**, select "Package version is immature". The description notes this policy "Detects 3rd party packages whose version release date is less than 14 days old".
- Under **Actions & Notifications**, select "Block" to prevent these packages from being downloaded into your repositories.
- Click **Save Policy**

Step 2: Maintain Developer Velocity with Compliant Version Selection

Guideline: A developer is more inclined to bypass security gates if they slow down the pace of development. The secret to successful compliance is making the process as transparent to developers as possible.

Action: Instead of simply blocking a package, Curation uses a feature called **Compliant Version Selection** to provide a seamless developer experience. When an open source package that violates the "Immature Package Policy" is requested , Curation automatically tries to direct the package manager to the latest compliant version. The developer can accept the package or look for a suitable alternative. The bottom line is that developers get a secure, working package without interruption, and compliance is enforced and documented without the need for additional resources and IT support.

Step 3: Gain Full Visibility and Documentation with Audit Events

Guideline: A proactive policy is great, but you need a clear, auditable trail of every decision the system makes.

Action: Use the Audit Event function to look for specific packages and check the logs for suspicious activity. ThIs is essential for compliance, troubleshooting, and understanding the health of the software supply chain.

- Navigate to **All Projects > Curation > Audit Events.**
- Here, you can see a log of all packages and whether they were **Blocked, Approved,** or **Passed** based on your policies.
- You can easily filter to investigate specific packages. For example, after the npmincident, you could search for **chalk:5.6.1** to see its status and confirm it was blocked by your policies

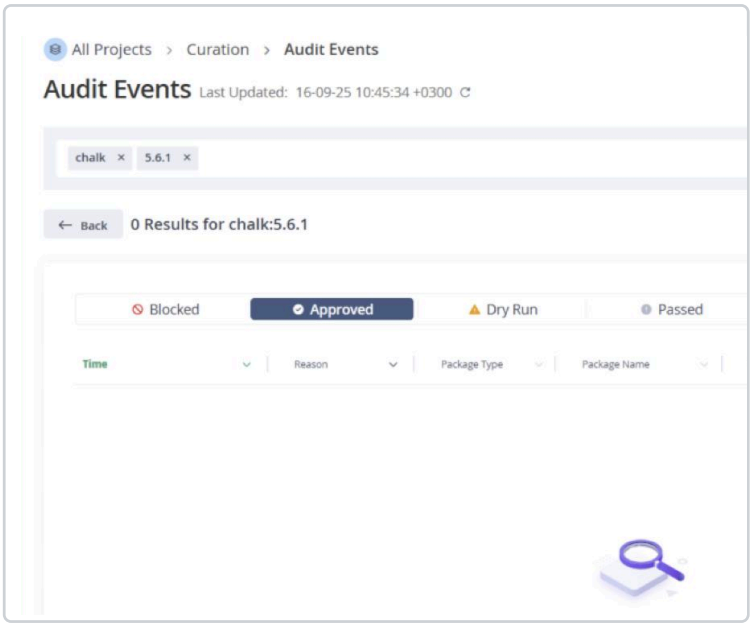

FIGURE 2: The Curation Audit Events screen provides a centralized log of all package evaluations.

BEYOND PREVENTION: A MULTI-LAYERED DEFENSE

JFrog Curation may be the first line of defense, but it takes the entire JFrog Platform to provide shift-left and shift-right security at every stage of the SDLC.

FIRST LAYER

Finding Existing Vulnerabilities with JFrog Xray: What if a malicious package was downloaded before the relevant Curation policies were in place? Xray continuously monitors your artifacts for newly discovered vulnerabilities and malicious code without needing disruptive rescans. You can use the **GET xray/api/v2/search/impactedResources** API to programmatically identify every resource impacted by a specific threat.

SECOND LAYER

Detecting Threats in Production with JFrog Runtime: Security doesn't stop at deployment. JFrog Runtime extends visibility into your production environments, where malicious packages are clearly identified in a live assessment screen, allowing you to see threats running in real-time.

THIRD LAYER

Remediating Potential Threats with JFrog Advanced Security: The platform also provides detailed remediation guidance. For a malicious package like chalk, this includes clear steps for removing the package, refreshing stolen credentials, stopping associated processes, and removing any installed backdoors.

FROM REACTIVE TO PROACTIVE SECURITY

The npm hijack attack was a wake-up call, proving that a reactive security posture is no longer viable. The key to upgrading software supply chain security is moving to a more proactive approach to minimize the attacker's window of opportunity.

By implementing automated controls with JFrog Curation, you can block risky and malicious packages before they enter your ecosystem, without sacrificing developer productivity. This is the essence of shifting "lefter than left." Following the npm incident, customers from highly regulated industries such as finance, reached out to accelerate their Curation deployment plans from next year to immediately. We even received direct feedback from customers saying that Curation **“saved my day”**.

EXTENDING BEYOND PACKAGES

Proactive security must extend beyond third party software packages to the entire software supply chain including development tools. Unvetted IDE extensions can introduce the same vulnerabilities as malicious code, but applying Curation's automated policies to these tools ensures an end-to-end trusted software development environment.

Ready to build your own proactive defense?

Then schedule a demo and see how the JFrog Platform can help protect you from present and future software supply chain attacks.