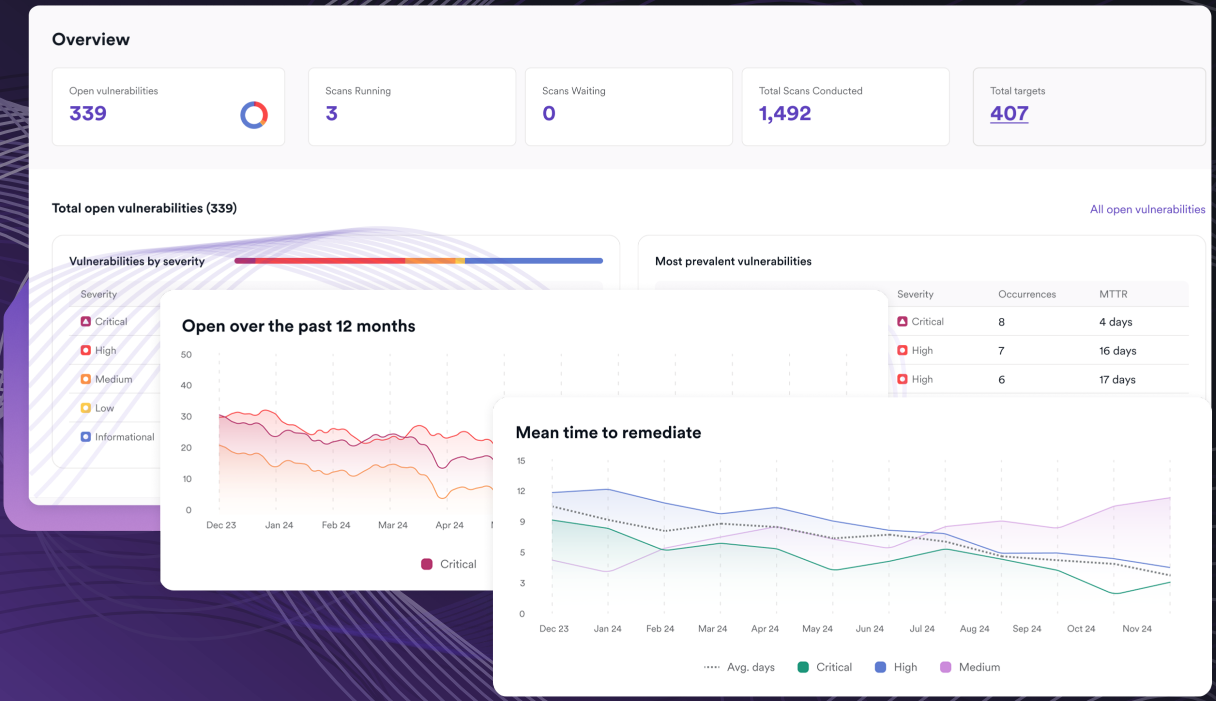
ASPM Guide

invicti

Protecting Your Apps Against Supply-Chain Threats Like React2Shell


Overview

Open vulnerabilities	Scans Running	Scans Waiting	Total Scans Conducted	Total targets
339	3	0	1,492	407

Total open vulnerabilities (339)

All open vulnerabilities

Vulnerabilities by severity

Severity
- Critical
- High
- Medium
- Low
- Informational

Most prevalent vulnerabilities

Severity	Occurrences	MTTR
Critical	8	4 days
High	7	16 days
High	6	17 days

Table of contents

Introduction

The React2Shell vulnerability that surfaced in early December 2025 is only the latest in a long series of high-profile software supply-chain flaws in recent years. Like Log4Shell in 2021 or Spring4Shell in 2022, it affects a widely-used package rather than one specific application. Such weaknesses exploit trusted open-source dependencies, spread rapidly through automated build systems, and enable devastating remote code execution with little or no authentication required.

When a new global threat like this hits, chaos is often the default response – unless you already have an organized and centralized system in place and set up to prevent exposure.

The system that can do this is an application security posture management (ASPM) platform that gives security and development teams the visibility, automation, and enforcement they need to respond quickly, remediate thoroughly, and prevent recurrence.

Here is a step-by-step playbook that ASPM users, and Invicti ASPM users in particular, can use to head off supply-chain security threats like React2Shell or Log4Shell and build a reliable system to future-proof their application environments.

Step-by-step response and prevention with Invicti ASPM

The moment a new vulnerability is disclosed, the first question is always: Are we affected? All too often, organizations have no immediate and accurate answer, especially when it comes to making use of vulnerable components. The initial disclosure of React2Shell sent security and engineering departments worldwide scrambling to answer the deceptively simple question: Are we using a vulnerable version of React anywhere in the company?

Step 1

Exposure discovery with SBOM Radar

Getting immediate and reliable answers starts with setting up and using software bills of materials (SBOMs) that list all the components used by an application. With Invicti ASPM, you can generate and maintain SBOMs for applications and services as part of security scans that are executed within the platform using the commercial or open-source tools of your choice. The SBOM Radar feature automatically cross-references your SBOM against the latest data in public vulnerability databases so you don't have to run new scans to find new vulnerabilities.

Following an initial scan, your SBOM is automatically updated with the latest vulnerability data every 12 hours, regardless of the last time you scanned. Within minutes, you can see exactly which builds, repositories, and production environments contain bad components that you need to look at – anything from React Server Component or Next.js versions affected by React2Shell to malware-laced npm packages affected by [Shai-Hulud]().

Any projects that have never been scanned can be easily triggered in bulk to ensure there is nothing that falls through the cracks. The Asset Management tool is designed specifically for this purpose.

With all those results in place, you can filter by environment, business criticality, internet exposure, or data classification to know what must be fixed first.

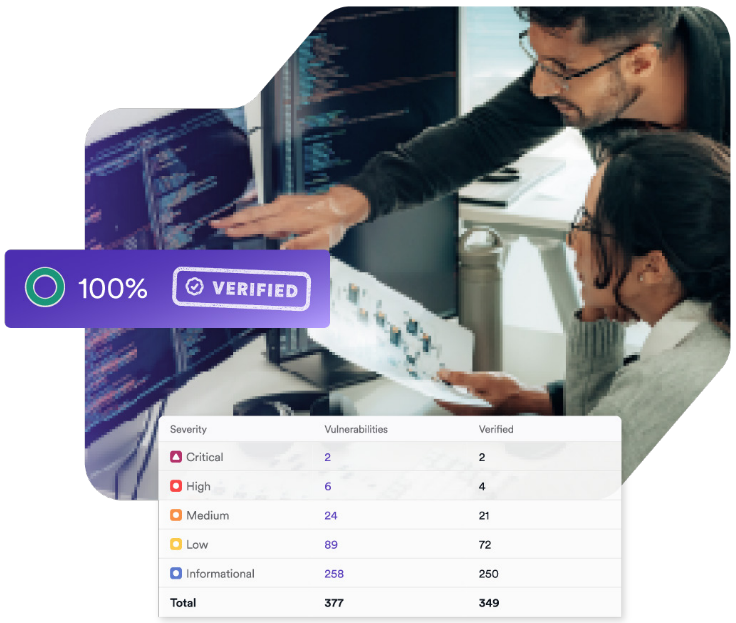
The result is a precise, prioritized list of exposed assets that automatically updates and alerts you to new risks as they emerge. No more blind scrambling or guesswork once exploitation is already widespread – now you're the first to know what needs fixing.

Step 2

Centralized triage and orchestrated remediation

After identifying where vulnerable components appear, the next step is organizing all related findings so you can remediate efficiently. Invicti ASPM provides a single place to aggregate and normalize results from SCA, proof-based DAST, container scanning, and other tools that may detect React2Shell-affected versions or related weaknesses. This eliminates conflicting severities, duplicate issues, and manual reconciliation.

To streamline response, the platform automatically applies your routing rules so each issue goes to the right repository or service owner with the relevant context included. For DAST findings, this often includes proof-of-exploit, which helps developers understand exactly why the issue matters. Policy-based triage rules let you escalate, suppress, or auto-close issues based on criteria like environment, business criticality, or exploit status, keeping teams focused on the work that actually reduces risk.



Severity	Vulnerabilities	Verified
Critical	2	2
High	6	4
Medium	24	21
Low	89	72
Informational	258	250
Total	377	349

Because all affected components and applications now appear in a central view, you can track remediation progress without spreadsheets or cross-team check-ins. As fixes land, Invicti ASPM updates the exposure status automatically based on incoming scanner data and SBOM changes.

The result is a structured, predictable workflow for driving fixes during a supply-chain event, with a full audit trail.

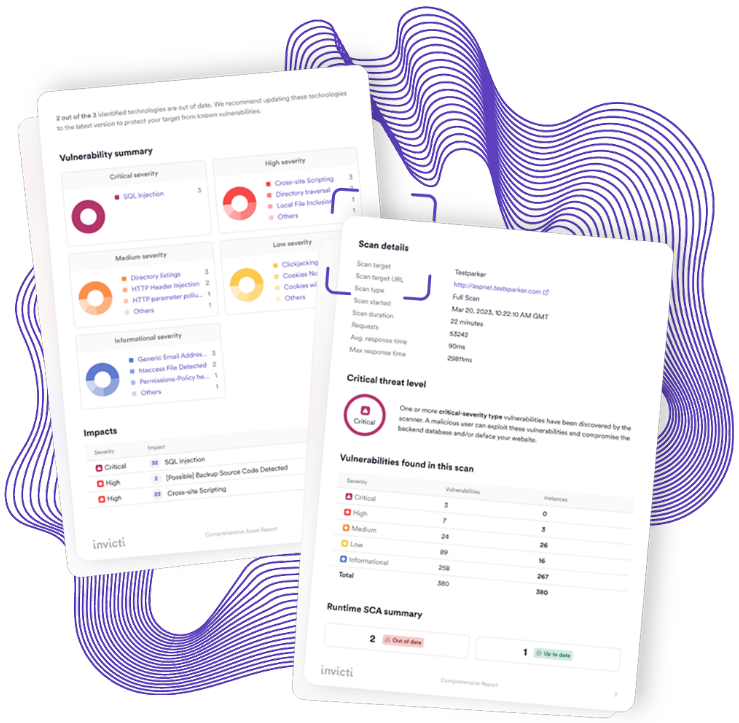
Step 3

Verifying fixes and checking for signs of compromise

Once patches or dependency updates are applied, you need to confirm that the vulnerable component is gone or that the fix behaves as expected. From Invicti ASPM, you can trigger proof-based rescans for any affected applications. When available, proof provides high-confidence confirmation that the exploit path is no longer accessible, which speeds up verification and reduces rework.

SBOM Radar adds another checkpoint by continuously re-evaluating your software inventory against updated vulnerability data. If a vulnerable version reappears because of dependency drift, a stale container image, or an overlooked service, you will see it immediately without rerunning scans.

If official advisories recommend checking for indicators of compromise, ASPM helps narrow the scope: you can see exactly which applications, builds, and environments were using the affected component and when. This provides a clear starting point for any manual investigation.



2 out of the 3 identified technologies are out of date. We recommend updating these technologies to the latest version to protect your target from known vulnerabilities.

Vulnerability summary

Critical severity	
SQL Injection	3

High severity	
Cross-site Scripting	3
Directory traversal	[illegible]
Local File Inclusion	1
Others	1

Medium severity	
Directory listings	3
HTTP Header Injection	2
HTTP parameter pollu...	1
Others	1

Low severity
Clickjacking
Cookies No
Cookies wi
Others

Informational severity	
Generic Email Addres...	3
htaccess File Detected	2
Permissions-Policy he...	1
Others	1

Impacts

Severity		Impact
Critical	32	SQL Injection
High	2	[Possible] Backup Source Code Detected
High	22	Cross-site Scripting

invicti Comprehensive Asset Report

Scan details

Scan target	Testparker
Scan target URL	http://aspnet.testsparker.com
Scan type	Full Scan
Scan started	Mar 20, 2023, 10:22:10 AM GMT
Scan duration	22 minutes
Requests	53242
Avg. response time	90ms
Max response time	29811ms

Critical threat level

Critical

One or more **critical-severity type** vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.

Vulnerabilities found in this scan

Severity	Vulnerabilities	Instances
Critical	3	0
High	7	3
Medium	24	26
Low	89	16
Informational	258	267
Total	380	380

Runtime SCA summary

2 Out of date

1 Up to date

invicti Comprehensive Report 2

Finally, the platform can generate reports showing the original finding, the fix event, and the validation result, simplifying internal review and external reporting.

This step gives you confidence that the fix worked and no attackers slipped through during the potential window of exposure. And for an n-day rather than zero-day vulnerability like React2Shell, where you can respond before working attacks emerge, you can eliminate your exposure by patching before exploitation even begins.

Step 4

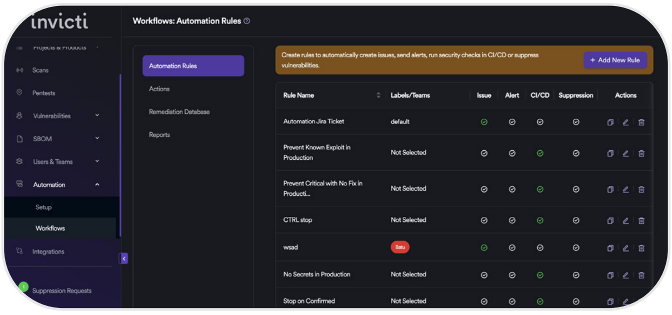
Lock the door with CI/CD enforcement rules

The crucial step for long-term security is making sure known-bad components can never make it back into your applications (or get into them in the first place).

This can cover vulnerable dependencies, as with React2Shell or Log4Shell, but it can also catch malware-laced packages like npm dependencies tainted by Shai-Hulud.

You can do this by defining mandatory CI/CD policies in Invicti ASPM:

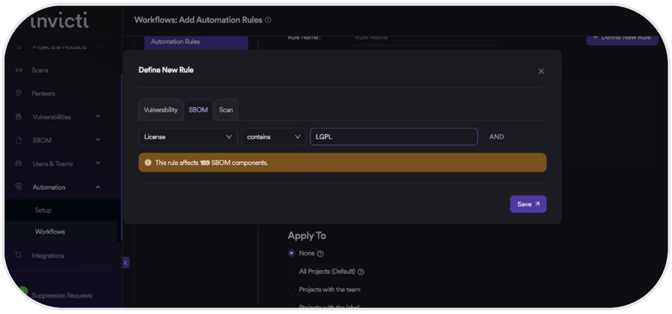
- Block builds that contain known-bad versions.
- Fail pipelines that attempt to install bad packages or their transitive dependencies.
- Optionally, use a tool like [kntrl] to enforce allow-list or deny-list rules in your CI/CD software, including GitHub Actions, GitLab CI, Jenkins, and Azure DevOps.



Workflows: Automation Rules

Create rules to automatically create issues, send alerts, run security checks in CI/CD or suppress vulnerabilities.

Rule Name	Labels/Teams	Issue	Alert	CI/CD	Suppression	Actions
Automation Jira Ticket	default	✓	✓	✓	✓	
Prevent Known Exploit in Production	Not Selected	✓	✓	✓	✓	
Prevent Critical with No Fix in Producti...	Not Selected	✓	✓	✓	✓	
CTRL stop	Not Selected	✓	✓	✓	✓	
wsad	Batu	✓	✓	✓	✓	
No Secrets in Production	Not Selected	✓	✓	✓	✓	
Stop on Confirmed	Not Selected	✓	✓	✓	✓	



Workflows: Add Automation Rules

Define New Rule

License contains LGPL AND

This rule affects 189 SBOM components.

Apply To

(●) None
( ) All Projects (Default)
( ) Projects with the team

From there, the SBOM Radar continuously monitors dependency drift in production and will alert you the moment a forbidden component reappears.

Building protection right into your pipeline minimizes the risk of regressions because bad components are blocked or eliminated long before they reach production.

Practical examples of setting up supply-chain security with Invicti

Using Software Bills of Materials (SBOMs)

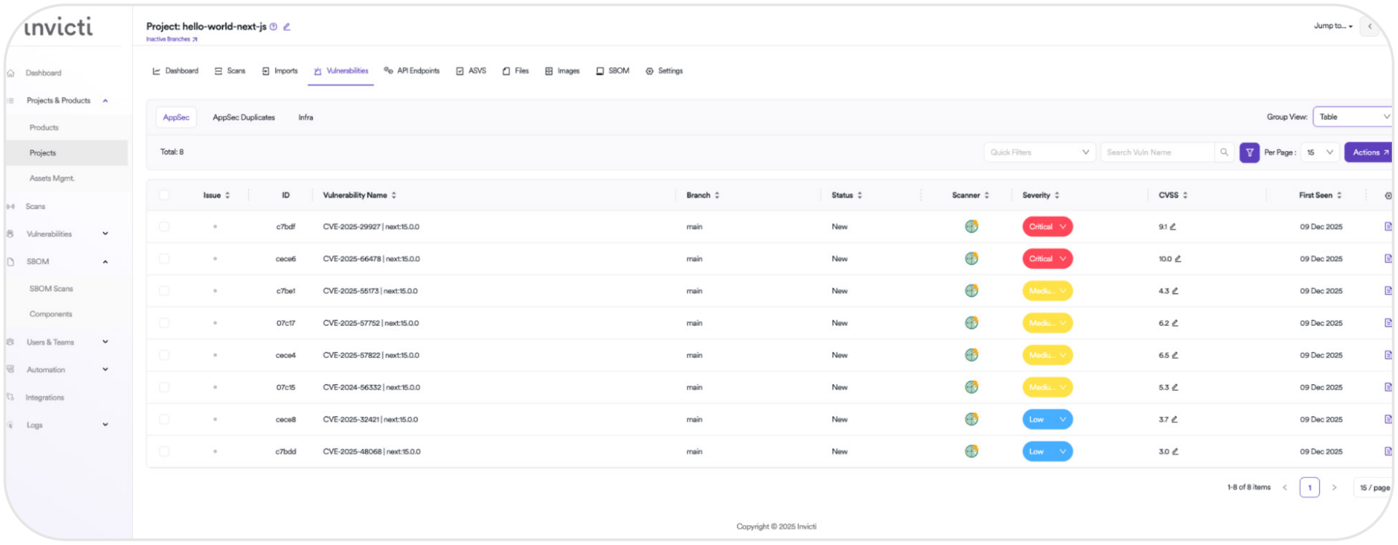
If you are systematically generating SBOMs and associating them with your deployed applications, your SBOM index can give you a full list of applications and versions that are in production. When a new vulnerability in a dependency like React or Next.js is announced, you can query your SBOM index to immediately identify every application that uses the affected version. This is critical for rapid patching and targeted remediation.

Uploading existing SBOMs into Invicti ASPM can also be done from the command line and automated using scripts. Here's an example of using the command line to import an SBOM for an application called *hello-world-next-js:*

```
$ kdt sbom import -p hello-world-next-js -f sbom-report.json -b main

2025-12-09T13:55:32 INF: sbom file imported successfully for:
[hello-world-next-js]
```

Invicti ASPM will automatically scan the uploaded SBOMs and store the SBOM list as follows:


Project: hello-world-next-js

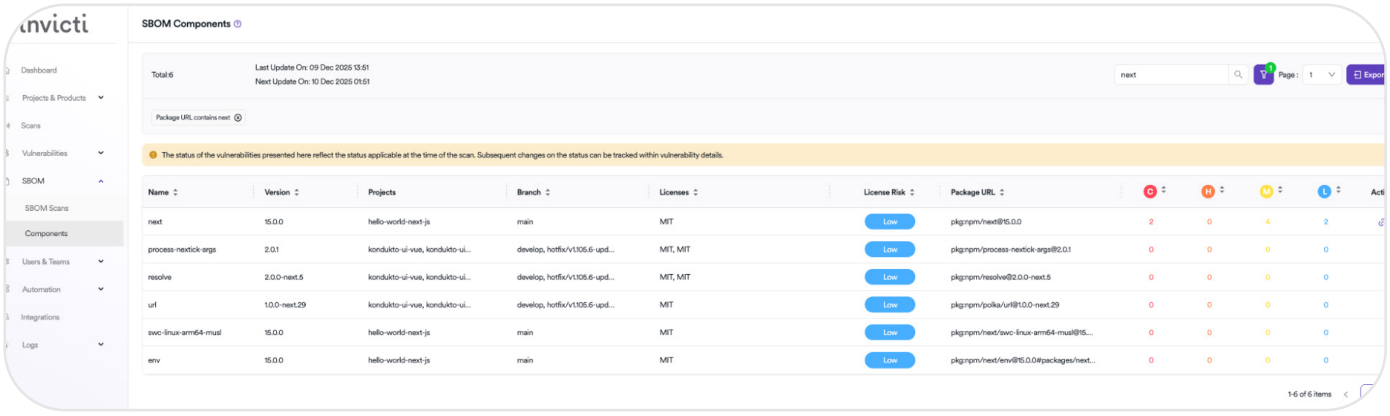
Total: 8

Issue	ID	Vulnerability Name	Branch	Status	Scanner	Severity	CVSS	First Seen
•	c7bdf	CVE-2025-29927 | next:15.0.0	main	New		Critical	9.1	09 Dec 2025
•	cece6	CVE-2025-66478 | next:15.0.0	main	New		Critical	10.0	09 Dec 2025
•	c7be1	CVE-2025-55173 | next:15.0.0	main	New		Mediu...	4.3	09 Dec 2025
•	07c17	CVE-2025-57752 | next:15.0.0	main	New		Mediu...	6.2	09 Dec 2025
•	cece4	CVE-2025-57822 | next:15.0.0	main	New		Mediu...	6.5	09 Dec 2025
•	07c15	CVE-2024-56332 | next:15.0.0	main	New		Mediu...	5.3	09 Dec 2025
•	cece8	CVE-2025-32421 | next:15.0.0	main	New		Low	3.7	09 Dec 2025
•	c7bdd	CVE-2025-48068 | next:15.0.0	main	New		Low	3.0	09 Dec 2025

1-8 of 8 items

Copyright © 2025 Invicti

Because Invicti ASPM stores all the uploaded SBOMs, you can then simply filter by the vulnerable package name and all the applications or projects that are using this package will appear:


SBOM Components

Total:6

Last Update On: 09 Dec 2025 13:51
Next Update On: 10 Dec 2025 01:51

Package URL contains next

The status of the vulnerabilities presented here reflect the status applicable at the time of the scan. Subsequent changes on the status can be tracked within vulnerability details.

Name	Version	Projects	Branch	Licenses	License Risk	Package URL	C	H	M	L
next	15.0.0	hello-world-next-js	main	MIT	Low	pkg:npm/next@15.0.0	2	0	4	2
process-nextick-args	2.0.1	kondukto-ui-vue, kondukto-ui...	develop, hotfix/v1.105.6-upd...	MIT, MIT	Low	pkg:npm/process-nextick-args@2.0.1	0	0	0	0
resolve	2.0.0-next.5	kondukto-ui-vue, kondukto-ui...	develop, hotfix/v1.105.6-upd...	MIT, MIT	Low	pkg:npm/resolve@2.0.0-next.5	0	0	0	0
url	1.0.0-next.29	kondukto-ui-vue, kondukto-ui...	develop, hotfix/v1.105.6-upd...	MIT	Low	pkg:npm/polka/url@1.0.0-next.29	0	0	0	0
swc-linux-arm64-musl	15.0.0	hello-world-next-js	main	MIT	Low	pkg:npm/next/swc-linux-arm64-musl@15...	0	0	0	0
env	15.0.0	hello-world-next-js	main	MIT	Low	pkg:npm/next/env@15.0.0#packages/next...	0	0	0	0

1-6 of 6 items

Integrating SCA scanning into CI/CD pipelines

Preventing vulnerable code from ever reaching your production environment is the most efficient security measure. This is where software composition analysis (SCA) tools are indispensable.

Running an SCA scan directly in the CI/CD pipeline is the best approach to preventing vulnerable application components from making it into production releases. SCA tools automatically check open-source libraries and packages against extensive vulnerability databases to flag components such as React or Next.js versions that are vulnerable to React2Shell. You can fail the build when a high-severity vulnerability is detected, which lets you enforce security policy early to dramatically reduce risk exposure.

Invicti ASPM can run a default scan after you upload an SBOM, or you can use a specific tool (like Google's osv-scanner) to orchestrate a scan operation for a given SBOM file. As you can see, the scan fails:

```
$ kdt scan -p hello-world-next-js -t osvscannersca -b main --params=
sbom-file-scan:true -f sbom-report.json

2025-12-09T16:41:44 INF: scan status is [waiting]
2025-12-09T16:41:54 INF: scan status is [running]
2025-12-09T16:42:34 INF: scan finished successfully

NAME                      ID                 BRANCH   META
----                      --                 ------   ----
OSV Scanner SCA Scan      6938274134aadf6e   main

TOOL                      CRIT     HIGH      MED      LOW      SCORE
----                      ----     ----      ---      ---      -----
osvscannersca             0        0         0        0        0

DATE
----
2025-12-09 16:41:44

2025-12-09T16:42:35 INF: failed to wait for scan to finish: scan failed
to pass release criteria: project does not pass release criteria due to
[SBOM] failure
```

You can also run an SCA scanner on your own and then push the results to Invicti ASPM:

```
$ osvs-scanner scan source -L bom.json --call-analysis=all --format=json
--output osv.results.json

Scanned bom.json file and found 29 packages

$ kdt scan -p hello-world-next-js -t osvscannersca -b main
-f osv.results.json

2025-12-09T16:52:31 INF: scan status is [retrieving-results]
2025-12-09T16:52:41 INF: scan finished successfully

NAME                      ID                  BRANCH   META
----                      --                  ------   ----
OSV Scanner SCA Scan      6938299f85128c27    main

TOOL                      CRIT      HIGH      MED      LOW      SCORE
----                      ----      ----      ---      ---      -----
osvscannersca             8         0         0        0        80

DATE
----
2025-12-09 16:52:31

2025-12-09T16:52:41 INF: failed to wait for scan to finish: scan failed
to pass release criteria: project does not pass release criteria due to
[SBOM] failure
```

When Invicti ASPM is set up as the release gatekeeper, it will block the release if a security scan fails. In this example, the release is stopped due to a failed SBOM check:

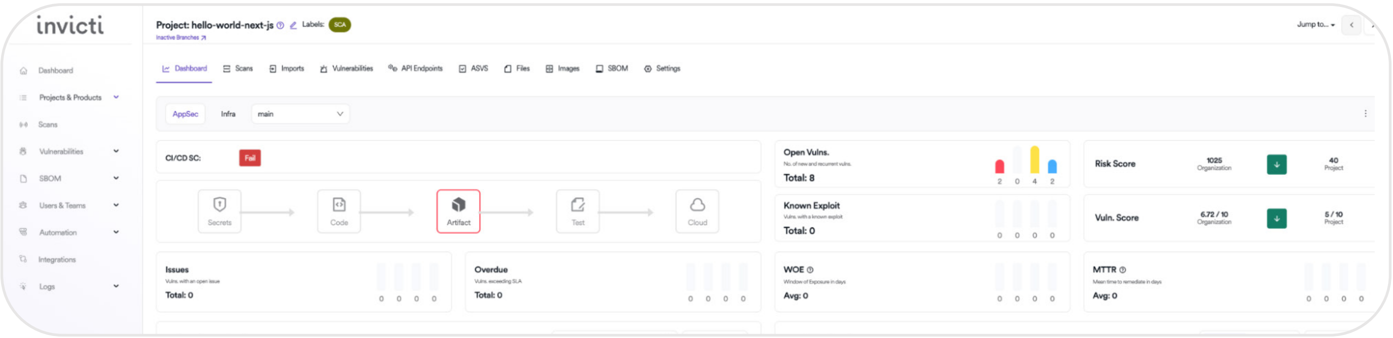
```
$ kdt release -p hello-world-next-js

STATUS        SAST          DAST          PENTEST       IAST
------        ----          ----          -------       ----
fail          undefined     undefined     undefined     undefined

SCA           CS            IAC           MAST          SBOM
---           --            ---           ----          ----
undefined     undefined     undefined     undefined     fail

2025-12-09T15:31:20 INF: project does not pass release criteria due to
[SBOM] failure
```

Of course, the same information can be viewed in the UI. Here’s a project/application view that shows there is a vulnerable artifact that causes the CI/CD pipeline to break:



Project: hello-world-next-js
Labels: SCA

CI/CD SC: Fail

Open Vulns. No. of new and recurrent vulns. Total: 8 — 2 0 4 2

Known Exploit Vulns. with a known exploit Total: 0 — 0 0 0 0

Risk Score 1025 Organization 40 Project

Vuln. Score 6.72 / 10 Organization 5 / 10 Project

Issues Vulns. with an open issue Total: 0 — 0 0 0 0

Overdue Vulns. exceeding SLA Total: 0 — 0 0 0 0

WOE Window of Exposure in days Avg: 0 — 0 0 0 0

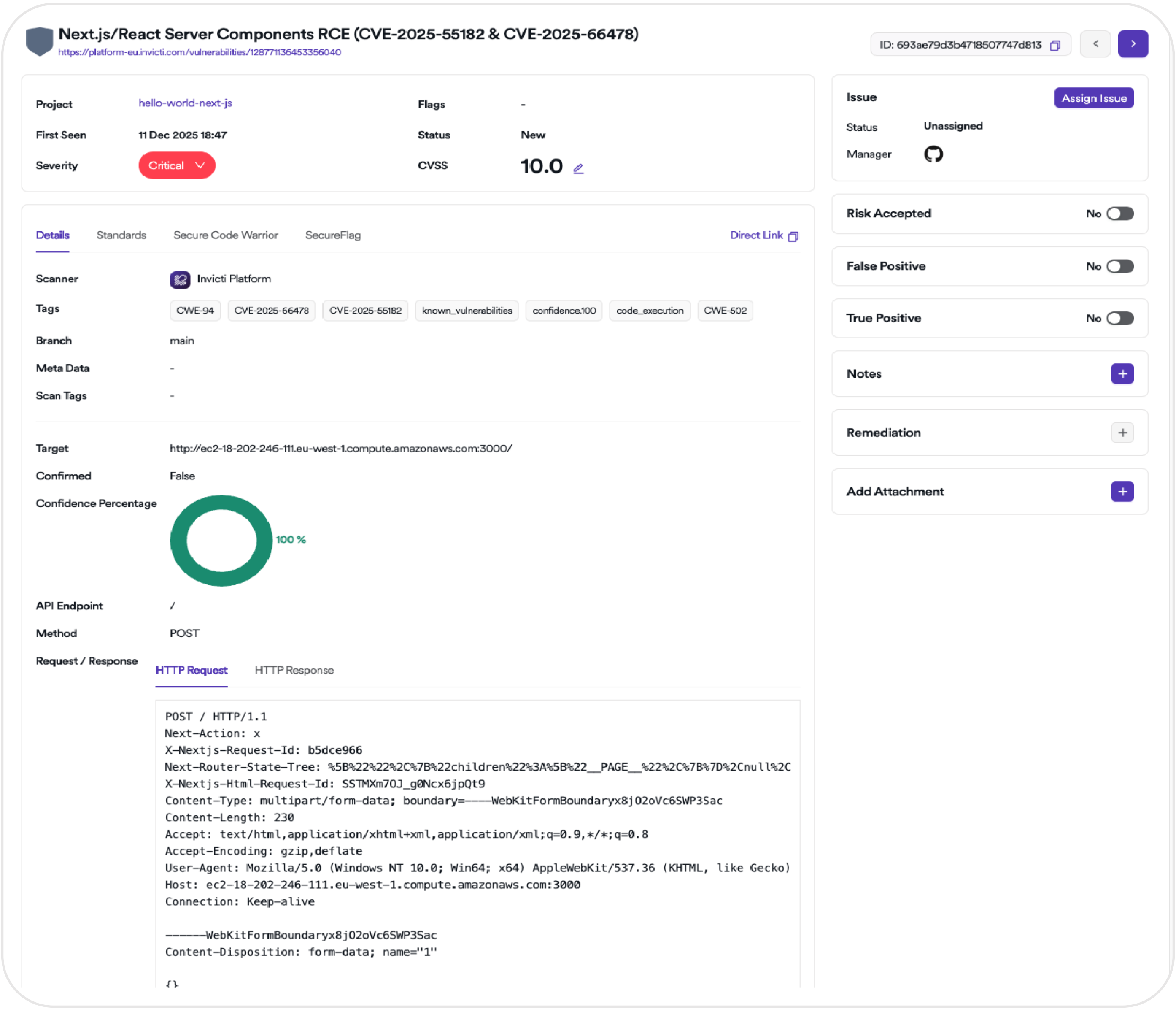
MTTR Mean time to remediate in days Avg: 0 — 0 0 0 0

Using proof-based DAST for detection and prioritization

Static SCA focuses on code components and known vulnerabilities, but dynamic application security testing (DAST) examines the running application from an attacker’s perspective. Running a DAST scan can show you immediately exploitable endpoints that you need to prioritize.

For many vulnerabilities, including React2Shell, Invicti’s proof-based DAST tool will not only flag a security issue but also confirm if a vulnerable dependency is actually exposed through an application endpoint. This context is invaluable for your security and development teams as it allows them to focus their remediation efforts on flaws that pose the most immediate and direct threat to users or data.

Here is an example of Invicti DAST reporting React2Shell in a vulnerable application:



Next.js/React Server Components RCE (CVE-2025-55182 & CVE-2025-66478)
https://platform-eu.invicti.com/vulnerabilities/128771136453356040

ID: 693ae79d3b4718507747d813

Project	hello-world-next-js	Flags	-
First Seen	11 Dec 2025 18:47	Status	New
Severity	Critical	CVSS	10.0

Issue	Assign Issue
Status	Unassigned
Manager	
Risk Accepted	No
False Positive	No
True Positive	No
Notes	
Remediation	
Add Attachment	

Scanner	Invicti Platform
Tags	CWE-94, CVE-2025-66478, CVE-2025-55182, known_vulnerabilities, confidence.100, code_execution, CWE-502
Branch	main
Meta Data	-
Scan Tags	-
Target	http://ec2-18-202-246-111.eu-west-1.compute.amazonaws.com:3000/
Confirmed	False
Confidence Percentage	100 %
API Endpoint	/
Method	POST
Request / Response	HTTP Request / HTTP Response

```
POST / HTTP/1.1
Next-Action: x
X-Nextjs-Request-Id: b5dce966
Next-Router-State-Tree: %5B%22%22%2C%7B%22children%22%3A%5B%22__PAGE__%22%2C%7B%7D%2Cnull%2C
X-Nextjs-Html-Request-Id: SSTMXm7OJ_g0Ncx6jpQt9
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryx8jO2oVc6SWP3Sac
Content-Length: 230
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Host: ec2-18-202-246-111.eu-west-1.compute.amazonaws.com:3000
Connection: Keep-alive

------WebKitFormBoundaryx8jO2oVc6SWP3Sac
Content-Disposition: form-data; name="1"

{}
```

By implementing these three levels of checks in Invicti ASPM, you can establish a robust and deeply integrated defense against current and future supply-chain vulnerabilities:

- SBOM for inventory
- SCA for prevention in CI/CD
- DAST for detection and prioritization

Conclusion

Why ASPM is non-negotiable in the era of supply-chain attacks

It's no accident that threat actors are increasingly focusing on the software supply chain. By exploiting the trust in and reliance on common packages used in thousands of applications, they can massively amplify their reach and the return on their efforts. And even if your application as a whole was secure yesterday, it can still be attacked tomorrow if one of its many dependencies is compromised.

In this climate, organizations need centralized visibility into application security posture at every level, from components to frontends to exposed APIs. Without a unified ASPM platform, teams waste days or weeks correlating data from half a dozen disconnected tools, arguing over ownership, and praying they didn't miss an affected service during manual checks.

A centralized platform like Invicti ASPM replaces chaos with order by giving you:

- One source of truth for all application risk
- Full coverage and accuracy when isolating and mitigating supply chain issues
- Proactive visibility with SBOM Radar
- Automated workflows that scale across thousands of applications
- Hard enforcement gates that turn best practices into guaranteed behavior

The SBOM Radar and CI/CD enforcement steps described in this guide to help you contain React2Shell today are the same workflow that would have countered Log4Shell and Spring4Shell in past years. Most importantly, the same steps will prepare your organization for the next inevitable supply-chain threat when (not if) it arrives – and monitoring the entire application supply chain is fast becoming an AppSec necessity.

So don't wait for the next worm or "-Shell" to force a fire drill. Organize your defense now with Invicti ASPM and turn reactive panic into repeatable, calm confidence.

Not yet an Invicti customer? Schedule a demo today and discover how quickly you can achieve true application security posture management.