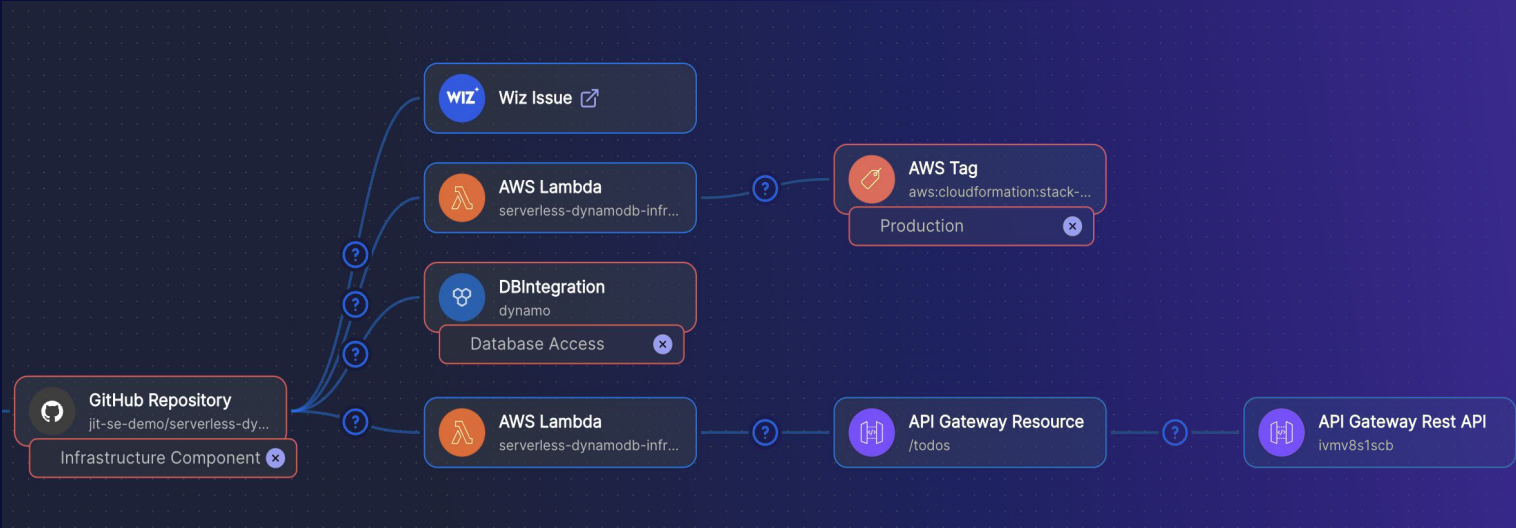




# Bridge the Gap Between ASPM and CNAPP with Jit + Wiz

Empower developers to consistently resolve security issues before production, while providing full code-to-cloud security coverage for security teams to prioritize real risks based on runtime context.



## Why we built the Jit + Wiz integration

Securing apps in the cloud has traditionally required implementing an array of code and cloud security scanners, including SAST, SCA, secrets detection, CSPM, and many others. There are many capable scanners that can detect code flaws and misconfigurations, but not many that can help developers consistently resolve real security issues before production – which is due to three core product security challenges:

### Lack of runtime context to prioritize real risks

Runtime context is needed to understand the true risk of security issues, including: whether issues reside in production or are exposed to the internet.

### Disjointed developer UX

Developers often resist code security scanners because they force developers outside of their environment and lack clear remediation guidance for security issues.

### Siloed vulnerability management & prioritization

Vulnerability prioritization is overcomplicated when there are many different tools prioritizing security findings in different ways.

Using the Jit + Wiz integration, security teams can unify product security findings and highlight the real risks with contextual prioritization, while empowering developers to independently resolve security issues without slowing them down.

# Jit + Wiz: The easiest way to unify, prioritize, and resolve product security issues in the cloud

The bidirectional integration makes it easy for developers to consistently resolve security issues **before** production, and for security teams to unify and prioritize the top risks **in** production – effectively bridging the gap between the core objectives of ASPM and CNAPP.

**Empower developers to consistently resolve issues before production**

Provide immediate feedback on the security of every code change within the developer environment. Using thorough remediation guidance and one-click issue resolution, developers don't need to be security experts to consistently resolve issues before production.

**Unified product security risk management & prioritization**

Push Jit's SAST findings into Wiz for a consolidated view of product security risk. Push Wiz findings into Jit to view security findings per repository.

**Enhance runtime context to focus on the real risks**

Stay focused on the real risks by combining the runtime context of both Jit and Wiz, providing deep insights into the impact of each security issue. Prioritize security findings that are deployed to production, exposed to the internet, and are connected to a database – among other prioritization factors.



**Developers understand each issue's impact and resolves them within their environment**

**Suggestion guidelines**

This remediation replaces the usage of insecure methods like 'innerHTML', 'outerHTML' or 'document.write' with a safer alternative, 'textContent'. The code will now use 'textContent' to safely set or update the content without putting your application at risk of XSS attacks.

Suggested change

```

6 - document.getElementById('content').innerHTML = userInput;
6 + document.getElementById('content').textContent = userInput;

```


Commit suggestion
Add suggestion to batch

**Why should you fix this issue?**

This code introduces a **vulnerability** that could impact **infrastructure** in a **production environment** that is also **externally accessible**. The public exposure of critical infrastructure increases the risk, and exploiting this vulnerability could lead to widespread disruptions or unauthorized control over the system.



**Enrich Wiz graphs with Jit-detected SAST issues to unify all product security risks**


**CWE-79**  
Jit Finding

Share
Star
More

**Description**

The vulnerability was discovered in file `src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js`, lines 1189–1189 .

User controlled data in methods like `innerHTML` , `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

**Name** CWE-79

**Project** Jit

**Severity** High

[View Details >](#)