



# GitLab

---

Integrating security into DevOps  
Aka DevSecOps

# Cindy Blake, CISSP

Product Marketing,  
Secure & Defend, GitLab



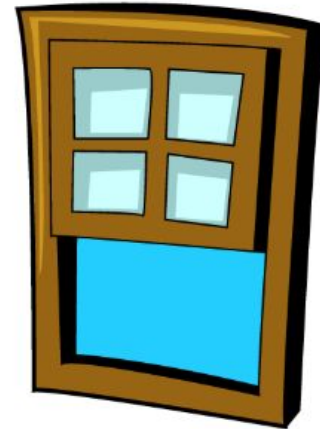
 [linkedin.com/in/cblake2000](https://www.linkedin.com/in/cblake2000)

 [@cblake2000](https://twitter.com/cblake2000)

 [gitlab.com/cblake](https://gitlab.com/cblake)

- IT Director, Developer, Strategic Marketing, Product Marketing
- Security evangelist
- Based in Bryan, Texas

# Do you have traditional application security?





## Key trends that affect security programs:

- “Shift Left” to find/fix vulns earlier
- The need to scale the security program
- DevOps/Agile collaboration
- MVC and Iterative deployments
- Policy-driven automation
- Serverless, cloud-native

## Challenges traditional app sec:

- Security teams have separate tools/processes and lack SDLC visibility
- Tests occur after individual code changes are merged with others’
- Scans test the entire application or the repo (out of context)
- Focus is on the app, what about IaC?

# Executive Order on Improving the Nation's Cybersecurity



**Biden Order To Require New Cybersecurity Standards In Response To SolarWinds Attack**

- Notification required
- Threat intel sharing
- Software Bill of Materials
- Modernization
  - Zero Trust architectures
  - Cloud services
  - Data analytics



- Greater efficiencies for both security and dev via
  - Collaboration
  - streamlined processes for both
  - Single source of truth
- Consistent compliance to policy
  - Cleaner and easier audits
  - Automated and consist use
  - Able to assess drift
- Reduced security exposure
  - Greater visibility to application risks, earlier
- Predictable costs
  - Security scans
  - Project time/budgets





“Your most important security product won’t be a security product.”

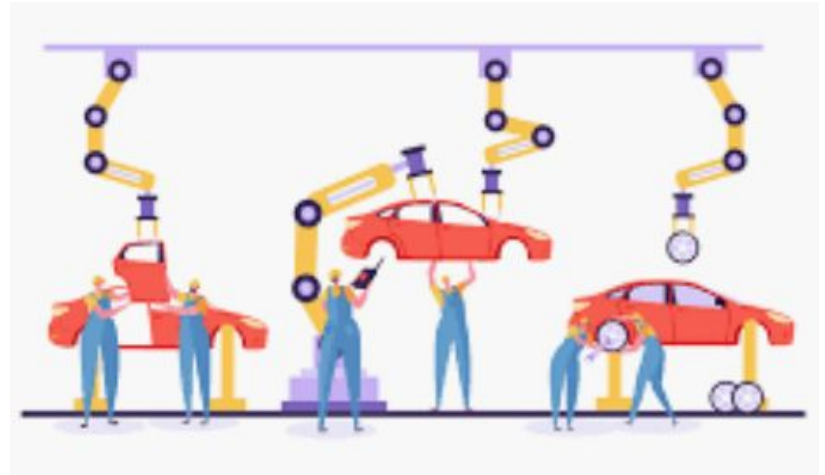
CISO of VMWare



## New problems need new solutions

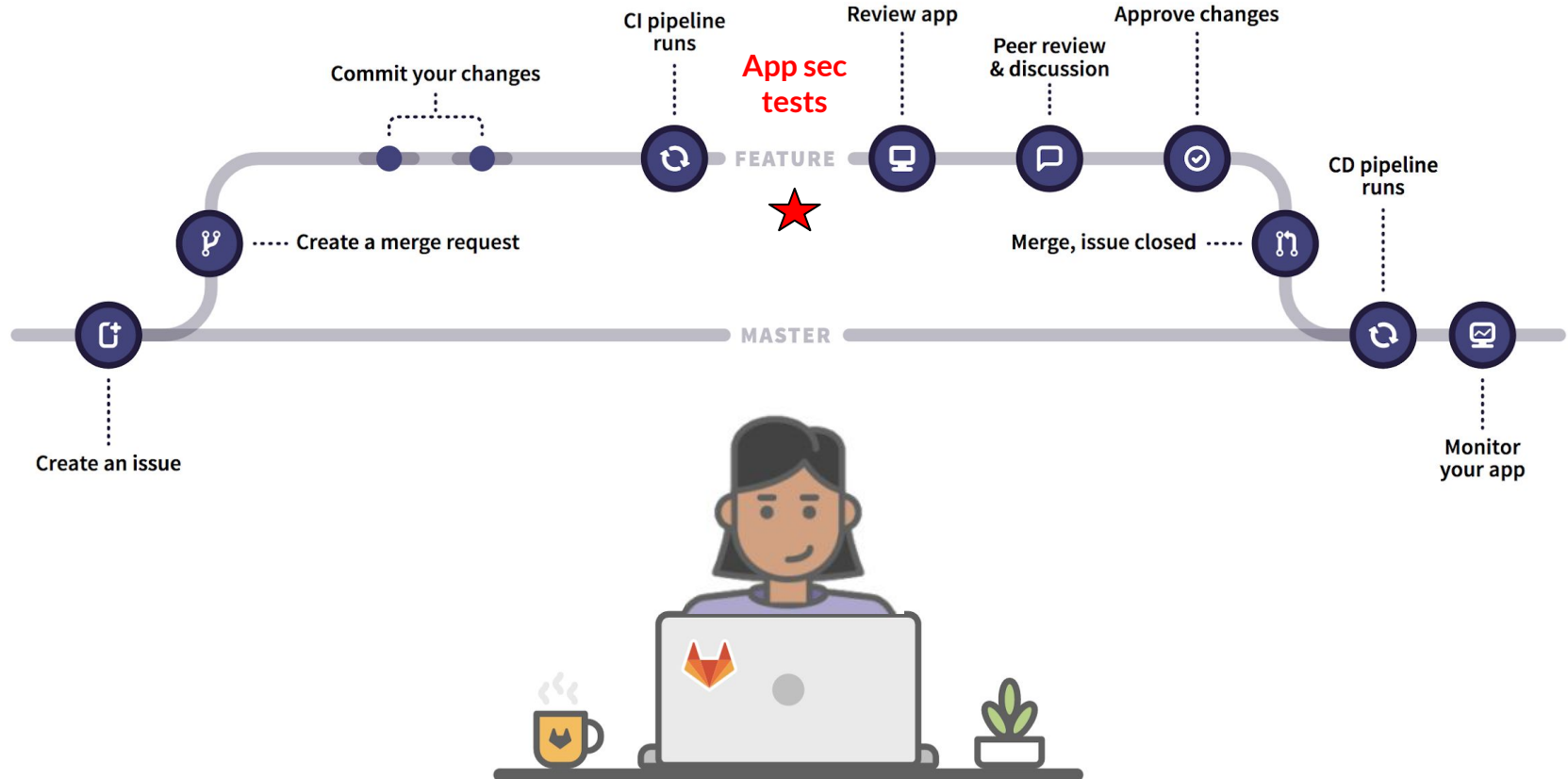
The CI pipeline is your software assembly line. It must be:

- Standardized
- Protected
- Measured
- Inspected





# GitLab seamlessly tests for vulnerabilities within the developer workflow



# Secure scanners today



## SAST

Static Application Security Testing scans the application source code and binaries to check for weaknesses and vulnerabilities.

## DAST

Dynamic Application Security Testing analyzes your running web application for known runtime vulnerabilities leveraging the Review App.

## Dependency Scanning

Analyze external dependencies (e.g. libraries like Ruby gems) for known vulnerabilities on each code commit with GitLab CI/CD.

## Vulnerability Management

View, triage, trend, track, and resolve vulnerabilities detected in applications giving you full visibility to your organization's risk.

## Secret Detection

Check for credentials and secrets in code commits and project history for allowing you to proactively resolve prior improper disclosure.

## Container Scanning

Analyze your containers for known security vulnerabilities in the application environment leveraging public vulnerability databases.

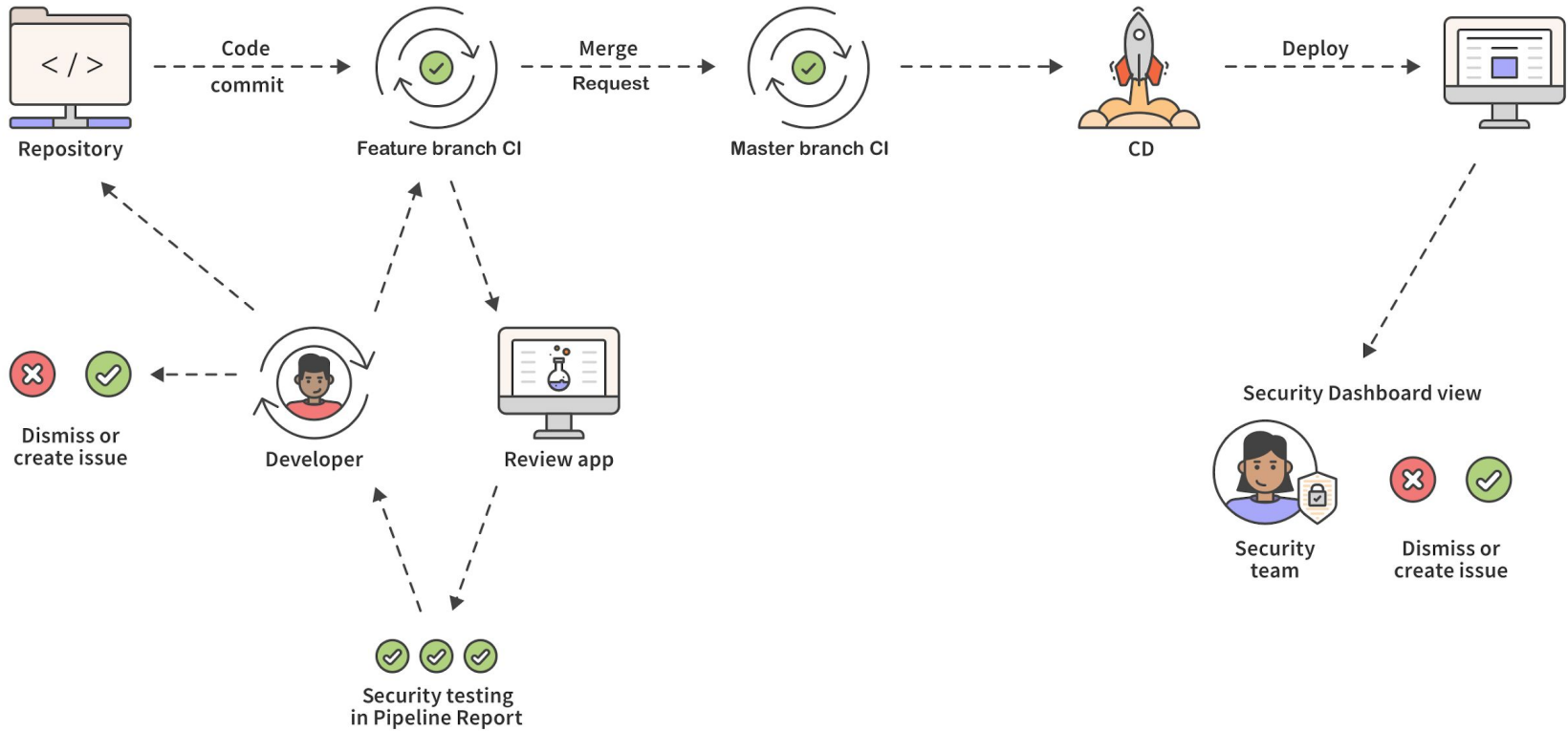
## License Compliance

Upon commit, project dependencies are searched for approved and blacklisted licenses defined by custom policies per project.

## Fuzz Testing

Input unexpected and malformed data into your application to measure response and stability with the goal of finding unknown vulnerabilities.

# Automate, allowing security to focus on exceptions



# Enabling Developer To Address Security Findings



## WIP: Feature Branch (to demonstrate MR widgets)

Overview 1 Commits 5 Pipelines 16 Changes 6

0/1 thread resolved



Request to merge `feature-branch` into `master`

The source branch is 7 commits behind the target branch

Open in Web IDE

Check out branch



Pipeline #90341293 passed for `d82a35f1` on `feature-branch`



Approve

Requires approval from Vulnerability-Check.



View eligible approvers



Security scanning detected 24 new, and 4 dismissed vulnerabilities for the source branch only

View full report

Expand



License Compliance detected 2 new licenses; approval required

Manage licenses

View full report

Expand

When vulnerabilities are present in an MR, you can easily **see** and **triage** them before the MR moves forward.

# ..in a developer-friendly way...with drill-down capabilities



The screenshot displays the GitLab interface for a pipeline run. The top navigation bar includes the GitLab logo, 'Projects', 'Groups', 'Activity', 'Milestones', and 'Snippets'. A search bar and notification icons are also present. The main content area shows the following details:

- Pipeline #26848268 passed for e4649a35 on feature-branch** (indicated by two green checkmarks).
- Add approval** button with the text: "No Approval required; you can still approve".
- Security scanning detected 63 new vulnerabilities and 3 fixed vulnerabilities**. Includes a "View full report" link and a "Collapse" button.
- SAST detected 59 new vulnerabilities and 1 fixed vulnerability**.
- A list of vulnerabilities:
  - Medium (Medium): Predictable pseudorandom number generator in `maven/src/main/java/com/gitlab/security_products/tests/App.java:47`
  - Medium (High): Cipher with no integrity in `gradle/src/main/java/com/gitlab/security_products/tests/App.java:29`
  - Medium (High): Use of insecure MD2, MD4, or MD5 hash function. in `python/imports/imports-aliases.py:11`
  - Medium (High): Use of insecure MD2, MD4, or MD5 hash function. in `python/imports/imports-aliases.py:12`
  - Medium (High): Use of insecure MD2, MD4, or MD5 hash function. in `python/imports/imports-aliases.py:13`
- Dependency scanning detected 2 new vulnerabilities and 1 fixed vulnerability**.
- A list of dependency vulnerabilities:
  - Unknown: Remote command execution due to flaw in the `includeParams` attribute of URL and Anchor tags for `org.apache.struts/struts2-core` in `pom.xml`
  - Unknown: Arbitrary OGNL code execution via unsanitized wildcard matching for `org.apache.struts/struts2-core` in `pom.xml`
  - Unknown: CSRF protection bypass for `org.apache.struts/struts2-core` in `pom.xml` (marked with a green checkmark)
- Container scanning detected 1 new vulnerability and 1 fixed vulnerability**.



## Cipher with no integrity ✕

**Identifiers:** [Find Security Bugs-CIPHER\\_INTEGRITY](#)

**File:** [gradle/src/main/java/com/gitlab/security\\_products/tests/App.java](#)

**Class:** com.gitlab.security\_products.tests.App

**Method:** insecureCypher

**Severity:** Medium

**Confidence:** High

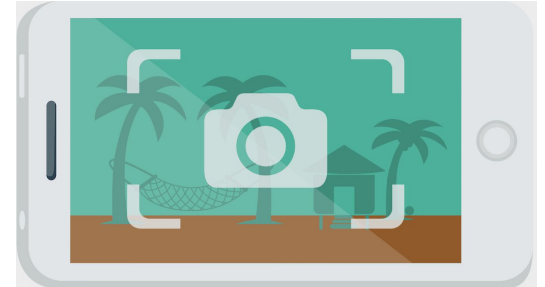
[Learn more about interacting with security reports \(Alpha\).](#)

# Advantage of this approach

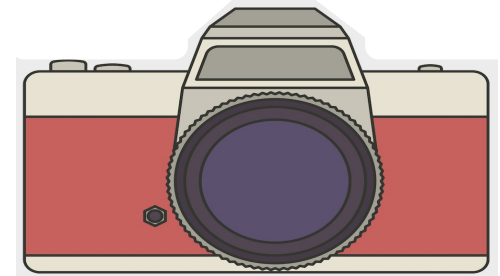


- Contextual
  - Within CI/CD dev workflow - accountable person
  - MR pipeline for dev
  - Security dashboard for Security
- Congruent with DevOps processes
  - Iterative within dev, tests every code change
  - Immediate cause/effect of code changes
- Integrated with DevOps tools
  - Create issues
  - Auto remediation
  - Production feedback
- Efficient and automated
  - Eliminate work wherever possible
  - No context-switching
  - Less tracking/triaging and more value-added security

## Simplicity and Integration Wins!



VS



# Providing Visibility Into Security Risk



## Quickly understand your at risk projects with Project Security Grades

The screenshot shows the GitLab Security Dashboard. On the left, there's a table of vulnerabilities with columns for Severity, Vulnerability, and Confidence. The table lists several critical vulnerabilities, such as 'XSS in data-target property of scrollspy for bootstrap' and 'Errors unhandled'. In the center, there's a 'Vulnerabilities over time' chart showing the number of vulnerabilities from May 17th to today, categorized by severity: Critical (22), High (41), Medium (13), and Low (7). A red dashed box highlights a 'Project security status' widget within the dashboard, which is a zoomed-in view of the right-hand panel.

Severity	Vulnerability	Confidence
CRITICAL	XSS in data-target property of scrollspy for bootstrap GitLab.org / security-products / binaries	High
CRITICAL	Errors unhandled. GitLab.org / security-products / binaries	Low
CRITICAL	Prototype pollution attack for extend GitLab.org / security-products / binaries	-
CRITICAL	Render path contains parameter value GitLab.org / security-products / binaries	-
HIGH	Insecure variable usage GitLab.org / security-products / binaries	High
HIGH	Nokogiri gem, via libxml2, is affected by multiple vulnerabilities GitLab.org / security-products / binaries	High
HIGH	Possible SQL injection GitLab.org / security-products / binaries	Low
MEDIUM	Unescaped model attribute <a href="#">gitlab-ce#1234</a> GitLab.org / security-products / binaries	Medium
MEDIUM	Unescaped parameter value GitLab.org / security-products / binaries	Medium
MEDIUM	Possible command injection GitLab.org / security-products / binaries	Medium
MEDIUM	Incomplete or No Cache-control and Pragma HTTP Header Set	Low

The zoomed-in view of the 'Project security status' widget shows a list of projects scored by vulnerability severity type detected. The projects are categorized by grade (F, D, C, B, A) and the number of projects in each category. The grades are color-coded: F (red), D (orange), C (purple), B (grey), and A (green).

Grade	Number of Projects
F	2 projects
D	9 projects
C	27 projects
B	13 projects
A	48 projects

Manage Security Risk Globally



# Providing visibility into security risk at the Project level



GitLab Projects Groups More

Search or jump to...

simply-simple-notes

Project overview  
Repository  
Issues 13  
Merge Requests 3  
Requirements  
CI / CD  
Security & Compliance  
Security Dashboard  
On-demand Scans  
Dependency List  
License Compliance  
Threat Monitoring  
Configuration  
Operations  
Packages & Registries  
Analytics  
Wiki  
Snippets  
Members  
Settings

GitLab-examples > security > simply-simple-notes > Security Dashboard

### Vulnerabilities

Export

Severity	Count
Critical	2
High	11
Medium	16
Low	30
Info	4
Unknown	0

Status: Detected +1 more | Severity: All severities | Scanner: All scanners

Status	Severity	Description	Identifier	Scanner	
<input type="checkbox"/>	Confirmed	Critical	AWS API key <a href="#">#20</a>	Gitleaks rule ID AWS	SAST GitLab
<input type="checkbox"/>	Detected	Critical	AWS API key <a href="#">#20</a> Remediated: needs review	Gitleaks rule ID AWS	SAST GitLab
<input type="checkbox"/>	Detected	High	Sandbox Escape in Jinja2 <a href="#">#16</a> Remediated: needs review	CVE-2019-10906 + 1 more	Dependency Scanning
<input type="checkbox"/>	Detected	High	Uncontrolled Memory Consumption in Django <a href="#">#16</a> Remediated: needs review	CVE-2019-6975 + 1 more	Dependency Scanning
<input type="checkbox"/>	Confirmed	High	CVE-2019-14697 in musl <a href="#">#8</a> Remediated: needs review	CVE-2019-14697	Container Scanning GitLab
<input type="checkbox"/>	Detected	High	Denial of service in Flask <a href="#">#6</a> Remediated: needs review	CVE-2019-1010083 + 1 more	Dependency Scanning
<input type="checkbox"/>	Detected	Medium	Insecure HTTP Method - DELETE	CWE-200 + 2 more	DAST
<input type="checkbox"/>	Detected	Medium	HTTP Only Site	HTTP Only Site + 2 more	DAST

Project's current vulnerability state

Line of code where vulnerability resides

Scanner that identified the vulnerability

Industry or public reference identifier

Remediated vulnerability awaiting review



- Automated, consistent CI pipelines
- Earlier visibility into risks and their remediation
- Predictable costs, even when testing more code more ways
- Reduced friction with dev
- Comprehensive feedback to dev



# We can work with incumbents, or replace them



“GitLab Secure replaced Veracode, Checkmarx, and Fortify in my DevOps toolchain. GitLab scans faster, is more accurate, and doesn’t require my developers to learn new tools”  
- *Financial services organization*

“GitLab Secure enables us to ship faster. Our other scanner tools could take up to a day to finish scanning whereas GitLab scans finish as little a few minutes”  
- *Healthcare services organization*

“GitLab Secure gives us unlimited scanning capability and across our entire GitLab repo. This is obviously a very “shift-left” move as issues will be identified directly in the repo for review and triage. We will be able to get the most coverage this way and it limits the onboarding.”  
- *Grocery retailer*



**Customers are actively migrating to GitLab**



The secret to successfully integrating Security into DevOps:

**Scan all code, every time**

**Seamlessly for dev**

**Using FEWER tools**

**With Dev, Sec, and Ops on the same page**

**And happy compliance auditors**



# GitLab

---

Everyone Can Contribute!