Cheat sheet: 10 Bitbucket security tips

www.snyk.io

1. Never store credentials as code/ config in Bitbucket.

Some good practices:

1. Block sensitive data being pushed to Bitbucket by adding git-secrets or a git pre-commit hook.
2. Break the build using the same tools when necessary.
3. Audit for slipped secrets with GitRob or truffleHog.
4. Connect to secret managers like Vault using Adaptavist's ScriptRunner.

2. Removing sensitive data

If sensitive data makes it to a repo:

1. Invalidate tokens and passwords.
2. Remove the info and clear the Git history (force push rewrite history).
3. Assess impact of leaked private info.

ScriptRunner make these integrations simple.

3. Tightly control access

Failures in security are often the results of humans making poor decisions. Mandate the following practices for your contributors:

1. Require two-factor authentication for all your Bitbucket accounts.
2. Never let developers share Bitbucket accounts/passwords.
3. Properly secure any laptops/devices with access to your source code.
4. Diligently revoke access from Bitbucket users who are no longer working with you.

Manage team access to data. Give contributors only access to what they need to do their work.

4. Add a SECURITY.md file

You should include a SECURITY.md file that highlights security -related information for your project. This should contain:

Disclosure policy

Define the procedure that describes what a reporter needs to do in order to fully disclose a problem safely when a security issue is found, including who to contact and how. Consider HackerOne's community edition or simply a 'security@' email.

Security update policy

Define how you intend to update project users about new security vulnerabilities as they are found.

Security-related configuration

Define the settings that your project users should configure that impact the security posture of deploying this project, such as HTTPS, authorization and many others.

Known security gaps & future enhancements

These are security improvements you haven't gotten to yet.

5. Validate Bitbucket apps

Remember these apps are written by third-party developers, not Bitbucket. Validate:

1. The application access rights
2. The author/organization credibility
3. How good the app's security posture is—a breach gives attackers access to your code!

Monitor changes in (2) and (3) over time, and keep on top of administering your applications.

6. Get security tips as part of your workflow with code insights

Perform scans on all your open PRs using Bitbucket Code Insights. The Snyk integration provides detailed in-line annotations about the new vulnerabilities that each PR introduces.

7. Add security testing to PRs

Use Bitbucket hooks to check that your PRs don't introduce new vulnerabilities:

1. Snyk—dependency vuln testing
2. SonarCloud—code quality testing
3. CodeClimate—automated code reviews

8. Add security testing in your Bitbucket pipes

Add security-scanning pipes into your CI/CD flow to ensure your automated pipelines do not contain security regressions:

1. Snyk—dependency vuln testing
2. SonarCloud—code quality testing

9. Consider Bitbucket server

If you don't want anybody to have access to your code (even Atlassian), or if regulations require it, use the Bitbucket Server on-prem offering.

10. Rotate SSH keys and personal access tokens

Bitbucket access is typically done using SSH keys or personal user tokens (in lieu of a password, because you enabled 2FA!). But what happens if those tokens are stolen and you didn't know?

Be sure to refresh your keys and tokens periodically, mitigating any damage caused by keys that leaked out.