Cheat sheet: 8 Azure Repos security best practices

snyk

1. Never store credentials as code/config in Azure Repos.

- Block sensitive data being pushed to Azure Repos by adding git-secrets or a git pre-commit hook.
- Break the build using the same tools when necessary.
- Audit for slipped secrets with GitRob or truffleHog.
- Use Azure KeyVault to store your keys and secrets in a Vault.

2. Removing sensitive data

If sensitive data makes it to a repo after all:

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

3. Tightly control access

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

- Never let developers share Azure repos accounts/ passwords.
- Properly secure any laptops/devices with access to your source code.
- Diligently revoke access from Azure repos 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. Inform your project users that those security controls aren't in place, and perhaps suggest they contribute an implementation!

5. Use Personal Access Tokens (PATs)

Personal access tokens are alternate passwords that can be used to authenticate one to Azure DevOps as well as other Microsoft tools such as Visual Studio.

6. Provide granular permissions and groups for users

Following the rule of least privilege, ensure that contributors exist in the correct groups and therefore have the necessary permissions to work. Try to restrict administrative actions where possible.

Additionally, monitor changes in requirements as contributors leave or step back from the project.

7. Add security testing to PRs

Use Azure Repos hooks to check that your PRs don't introduce new vulnerabilities with Snyk.

8. Rotate SSH keys and personal access tokens

Azure Repos access can be done using SSH keys or personal user tokens (in lieu of a password). 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.

Edward Thompson
Microsoft
ethomson@microsoft.com

Simon Maple
Snyk
simon@snyk.io