

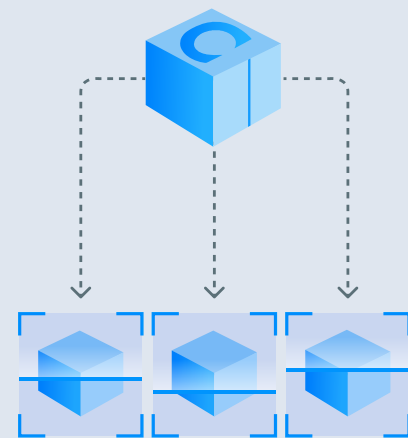
CONAN AUDIT CHEAT SHEET



CONAN
C/C++ Package Manager



WHAT IS CONAN AUDIT?



Conan is an open-source package manager for C and C++ applications that helps developers manage dependencies consistently across projects. It provides a flexible and reproducible way to handle libraries, ensuring compatibility across different platforms, compilers, and build systems.

Conan Audit is a Conan command that allows you to scan your C++ package dependencies for known vulnerabilities (CVEs). Based on the underlying engine that powers [JFrog Advanced Security](#), it delivers detailed vulnerability analysis reports, helping quickly identify and mitigate potential risks in dependency graphs.



INSTALLING CONAN AUDIT

Here are the steps for installing Conan Audit:

1. Install Conan
`pip install conan`
2. To register to the free service:
 - a. Visit <https://audit.conan.io/register> to sign up.
 - b. Fill in your details and submit the form.
 - c. You will receive a token and will receive an email to validate your token.
3. Use the token to authenticate the ConanCenter provider via an environment variable or through the command line:
`export CONAN_AUDIT_PROVIDER_TOKEN_CONANCENTER=<your token>`
`conan audit provider auth conancenter --token=<your_token>`
4. To check a single package
`conan audit list openssl/1.1.1w`
5. To scan the entire dependency graph of your project
`conan audit scan .`
6. You can also specify output formats, for example json or html:
`conan audit scan . --format=html > report.html`



KEY FEATURES



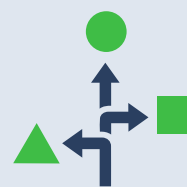
Vulnerability Detection

Conan Audit allows you to scan for known CVEs in any package that is available in Conan Center. When integrated with JFrog Advanced Security, Conan can provide even more granular vulnerability reports, including severity levels, CVSS scores, and remediation guidance. This helps developers quickly pinpoint potential issues before they can be exploited.



Transitive Dependency Analysis

Not only does Conan Audit inspect the target package, it also analyzes its entire dependency graph. This ensures that vulnerabilities in transitive dependencies are not overlooked, offering a complete picture of the security posture of the software supply chain.



Flexible Output Formats

Conan Audit supports multiple output formats such as JSON and HTML. Whether integrating vulnerability data into automated security pipelines or generating visual reports for compliance and management reviews, the flexibility of Conan Audit provides the right format for automation and compliance.



ENHANCED SECURITY INTEGRATIONS

Conan Audit integrates directly with [JFrog Catalog](#), for enhanced insights into potential security issues and enabling the setup of your own Catalog instance as the source for vulnerability information. Setup is simple and straightforward, by adding a private provider with the URL of your [JFrog Platform](#) instance and configuring the credentials.

Integration with [JFrog Artifactory](#) helps ensure that the components brought into your repositories are secure, while using JFrog Advanced Security together with Conan Audit enables a seamless shift left developer experience.



TOP 5 CONAN COMMANDS

1. List vulnerabilities for a single package

```
conan audit list <package_reference>
```

Example: Use this command to quickly check if a package has known vulnerabilities.

2. Scan the entire dependency graph

```
conan audit scan <path_to_project>
```

Example: Use this command to scan all dependencies in your project for vulnerabilities.

3. Authenticate the ConanCenter provider

```
conan audit provider auth conancenter  
--token=<your_token>
```

Example: Use this command to authenticate with the ConanCenter provider using your token.

4. Add your JFrog Catalog as a private security provider

```
conan audit provider add <provider_name> --type=private  
--url=<your_artifactory_url> --token=<your_token>
```

Example: Use this command to add your own private provider for enhanced security scanning.

5. List all available providers

```
conan audit provider list
```

Example: Use this command to get the list of all providers. Use the selected provider in audit list and audit scan by using the `--provider=<provider name>` argument.