snyk

Python Security Best Practices Cheat Sheet 2021

www.snyk.io

1. Always sanitize external data

Various issue types stem from unsanitized external data flowing into sinks. It is a best practice to sanitize external data close to the source.

2. Scan your code

There are a variety of tools available to scan your source code, but they work best when they integrate directly into your IDE.

3. Be careful when downloading packages

It's important that you can trust open source packages before adding them to your project. Check for the liveliness of the community and the security history of the package. Also, watch out for typosquatting when choosing a package.

4. Review your dependency licenses

Open source dependencies come with a license that could impact your project (and even force you to publish your code, too). Check on the license beforehand and make sure you comply.

5. Do not use a pre-installed Python version

Most distributions of Linux (and some generic container images) come with a Python version preinstalled. Unfortunately, the versions are typically not well-maintained, so it's a best practice to perform the specific installation yourself.

6. Use Python's capability for virtual environments

Python comes with a nice support of virtual environments that helps separate projects and their dependencies. It is a best practice to make use of this functionality. Most tools and IDEs support it out of the box.

7. Set `DEBUG = False` in production

During development, it is extremely helpful to have extended debugging messaging. Be sure to lower the output level post-development, as you don't want to allow this rich information out of your production environment.

8. Be careful with string formatting

Python apps suffer quite frequently from puzzling string encoding. Be careful with strings and think about their encoding when calling APIs or using them internally.

9. Deserialize very cautiously

Python is extremely versatile and it is possible to deserialize external data directly into internal class representations. This presents the danger of adding or changing classes or members with malicious content. Additionally, when deserializing, limit the size to prevent overloading.

10. Use Python type annotations

Python recently added the capability to add annotations to types. While this does not influence the interpreter, it can be used by static analysis tools. This is a best practice for all development, not just secure development.

Secure your Python code for free

Scan your Python code for quality and security issues and get fix advice right in your IDE.

Try Snyk