



# Python Code Audit

A modern Python source code analyzer based on distrust

# Proactive security by design

Cyber security breaches have a **severe** impact on our safety and privacy.

The majority security incidents are caused by **unknown** software vulnerabilities. But 90% of the security tools can only detect known vulnerabilities.

Don't just assume the Python code you run is secure and without design errors and coding mistakes.

**Validate it!**

Good effective security means preventive solutions. So [Shift Left](#).

*Python Code Audit is a modern tool for analyzing Python code and detecting potential security issues.*

# SAST (Static Application Security Testing) is crucial

Python is the most used programming language to date. Especially in the AI/ML ecosystem, most tools are based on Python. Large and small businesses use and trust Python to run their business. And for a reason: Python is also from a security perspective a good choice!

But even when using Python, the risk of security issues is still not zero. Running **insecure Python programs** can have disastrous consequences for your security and privacy.

To validate the Python code on possible security issues you should::

1. **Prevent** security issues when creating Python software and
2. **Inspect** downloaded Python software (packages, modules, etc) before running.

The best Python developers make mistakes. Detailed knowledge of the specific risks that can be introduced when using common Python library constructs is **not common knowledge**.

Secure programming is **difficult** and a **complex skill**. Secure programming requires expertise that goes far beyond Python programming. It requires in-depth security knowledge and experience and continuous learning.

**Using Python Code Audit during development saves time and increases the quality of programs created.**

# Python Code Audit capabilities

Python Code Audit is created to find **potential security issues** in Python code.

The static application security testing (SAST) tool 'Python Code Audit' has great features to simplify necessary security tasks:

- Detect and report potential vulnerabilities from all Python files in a directory. This is a must do check when researching python programs on possible security issues.
- A [cyclomatic complexity](#) count along with other relevant security statistics is determined per Python file. Statistics are aggregated over all files belonging to a program to give direct insights in possible security risks using simple visualizations.
- Detect and reports which Python modules are used per Python file. Known vulnerability information is directly reported.
- For every Python file, potential vulnerability issues are shown with the line number and code snippets that should be inspected or changed.
- Python Code audit has more than 62 security validations implemented to check potential security issues when using standard Python calls.
- All output is saved in simple static HTML reports. These reports can be examined in every browser.

# Installation

You can install Python CodeAudit via pip:

```
pip install -U codeaudit
```

This will install everything you need to directly use Python Code Audit!

Python Code Audit has extensive documentation to offer help on potential issues that are detected.

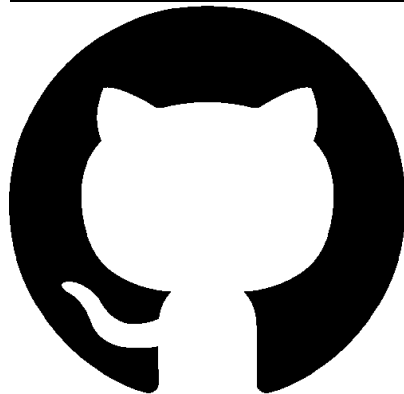
[Code Audit Documentation](#)

# Free and Open Source

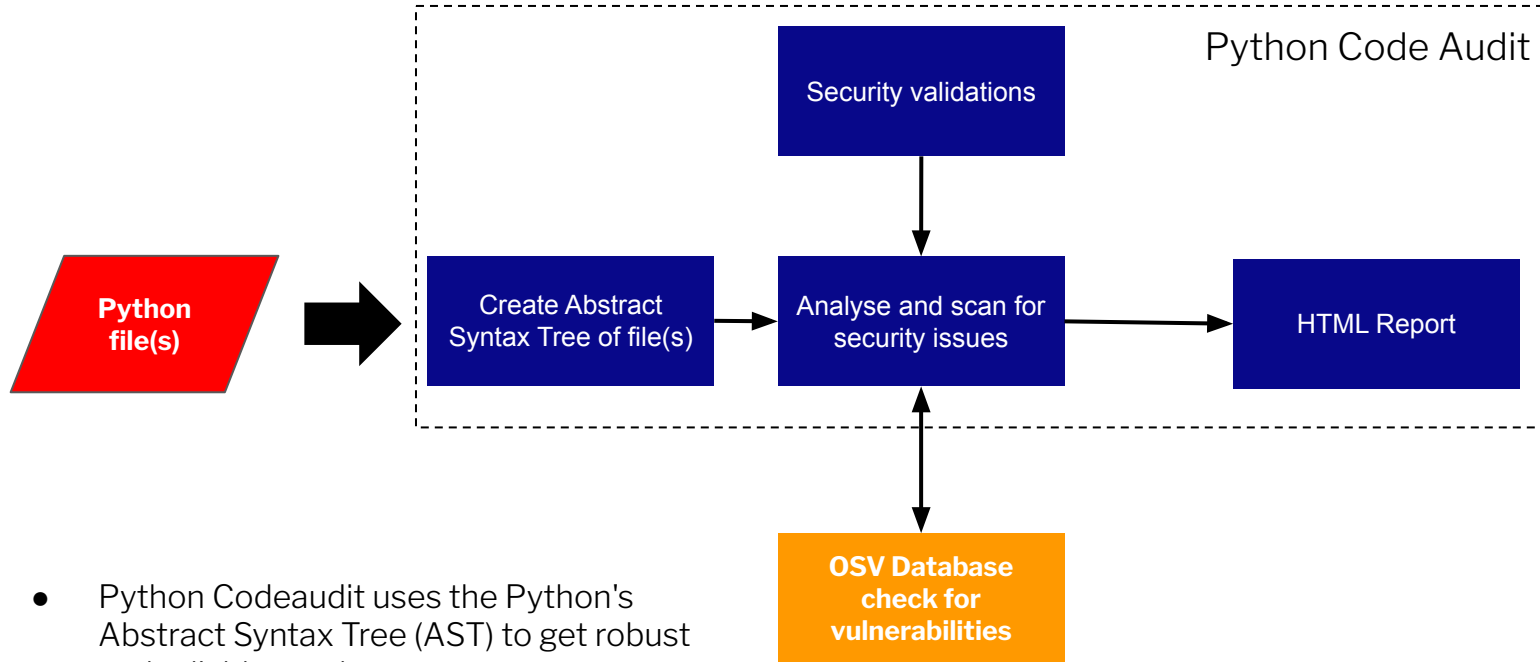
[Python Code Audit](#) is available under the GNU General Public License (GPL) license. Static Application Security Testing(SAST) for python code should not be a paid option. We believe in the power of FOSS, so [Python Code Audit](#) is a **local first** FOSS solution that anyone can use for free to make our digital world a bit more secure.



Source code on:



# Architecture overview



- Python Codeaudit uses the Python's Abstract Syntax Tree (AST) to get robust and reliable results.
- The [Open Source Vulnerability Database](#) is used for retrieving vulnerability information for external modules.

# Example of a report section (1)

## Codeaudit overview report

Codeaudit overview scan of the directory: `ncrypt/`

### Summary

| Number_Of_Files | Number_Of_Lines | AST_Nodes | Core Modules | External Modules | Functions | Classes | Comment_Lines | Median_Complexity | Maximum_Complexity |
|-----------------|-----------------|-----------|--------------|------------------|-----------|---------|---------------|-------------------|--------------------|
| 30              | 3188            | 1628      | 20           | 55               | 83        | 11      | 337           | 12.2              | 65                 |

Based on the maximum found complexity in a source file: Security concern rate is **HIGH**

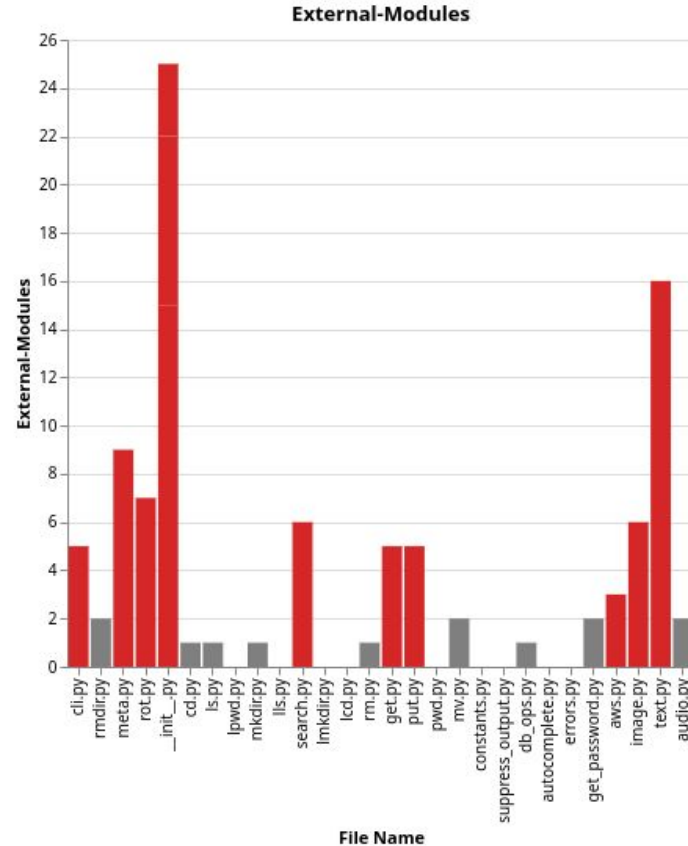
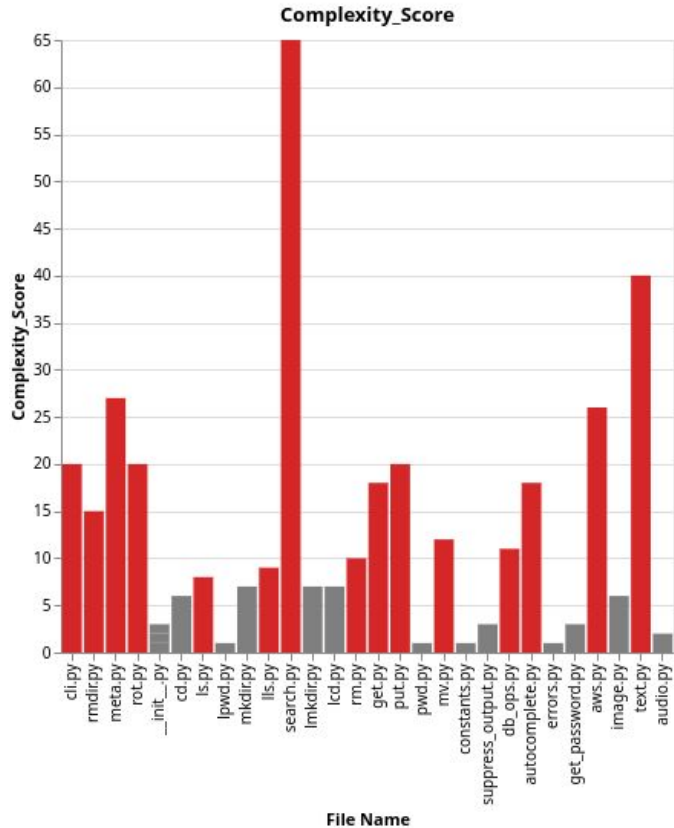
Based on the total Lines of Code (LoC) : Security concern rate is **HIGH**

► Click to see all discovered modules.

### Detailed overview per source file

► Click to see the report details.

# Example of a report section (2)



# Example of a report section (3)

## Codeaudit report

### Directory scan report

Below the result of the Codeaudit scan of the directory: `ncrypt/`

Total Python files found: 30

#### Result for file `cli.py`

Location of the file: `/home/maikel/tmp/ncrypt/ncrypt/cli.py`

Number of potential security issues found: 4

▼ Click to see the details for found security issues.

| line | validation  | severity | info   | code   |
|------|-------------|----------|--|--|
| 77   | os.makedirs | Low      | Operating System calls can have a security impact and should be inspected in detail. | <pre>os.makedirs(self.root_dir)</pre>                                |
| 89   | os.makedirs | Low      | Operating System calls can have a security impact and should be inspected in detail. | <pre>os.makedirs(os.path.join(self.root_dir, "local_files"))</pre>   |
| 94   | os.makedirs | Low      | Operating System calls can have a security impact and should be inspected in detail. | <pre>os.makedirs(self.key_dir)</pre>                                 |
| 128  | os.system   | High     | Operating System calls can have a security impact and should be inspected in detail. | <pre>try:<br/>os.system("cls" if os.name == "nt" else "clear")</pre> |

▶ Click to see details for file `/home/maikel/tmp/ncrypt/ncrypt/cli.py`

▶ Click to see details for used modules in this file.

# Join!

We believe that:

- Cyber security protection can be better and
- Cyber security solutions can be simpler.
- We should only use cyber security solutions that are transparent, and we can trust.

Openness is key. You don't have to be a genius to make cyber security simpler and better.

So Join the [community to contribute](#) to the most complete, local first , Python Security Code Scanner.

***Join the journey!***



## Python Codeaudit

A modern Python source code analyzer based on distrust

# Sponsors

## NO|Complexity

Complexity is easy, but simplicity is truly  
valuable



**Diagnosis, Design and Change**

Use external expertise when dealing with architecture and security. An objective external review is a must to prevent tunnel vision. Make use of the services offered from our sponsors!

As a free and open source project, you can [sponsor](#) Python Code Audit to increase the audience for your commercial offerings.