

[New issue](#)

# [Security Vulnerability] CWE - 94 Code Injection in python\_executor Class Due to Unvalidated exec() Usage #50

[Open](#)

ybdesire opened 2 weeks ago



## Description

The provided Python code contains a significant security vulnerability of type CWE - 94: Code Injection. The vulnerability exists because the code uses the `exec()` function to execute user - provided input without any form of validation.

Impacted code as below:

[factool/factool/math/tool.py](#)Line 15 in [e02bc8b](#)

```
15     exec(program)
```

[factool/factool/math/tool.py](#)Line 22 in [e02bc8b](#)

```
22     exec(program)
```

The `python_executor` class has a `run_single` method that takes a `program` parameter. This `program` is directly passed to the `exec()` function, which means any code within it will be executed. The same `program` is also executed a second time in the `run_single` method. Additionally, the `run` method calls `run_single` with a `snippet` parameter, which is presumably user input.

When an attacker provides malicious code as input, the `exec()` function will execute this code. This can lead to a wide range of security issues, such as unauthorized access to system resources, data leakage, or even complete system compromise. For example, an attacker could use the injected code to read sensitive files, modify system settings, or launch further attacks on the underlying infrastructure.

## Exploit

An attacker can exploit this vulnerability by providing malicious Python code as input to the `run` or `run_single` methods. Here are some examples of how an attacker could exploit this:

### 1. Reading Sensitive Files

The attacker could provide the following code as input:

```
with open('/etc/passwd', 'r') as f: print(f.read())
```



If the Python process has the necessary permissions, this code will read the contents of the `/etc/passwd` file, which contains user account information on Unix - like systems.

## 2. Executing System Commands

The attacker could use the `os.system` function to execute system commands. For example:

```
import os; os.system('ls -l')
```



This code will execute the `ls -l` command, which lists the files and directories in the current working directory. More malicious commands could be used, such as deleting files or creating backdoors.

## 3. Modifying System State

The attacker could also try to modify the system state by changing environment variables or system settings. For example:

```
import os; os.environ['PATH'] = '/malicious/path:' + os.environ['PATH']
```



This code modifies the `PATH` environment variable, which could cause the system to execute malicious binaries instead of legitimate ones.

In summary, the lack of input validation in the use of the `exec()` function makes this code highly vulnerable to code injection attacks.

[Sign up for free](#) to join this conversation on GitHub. Already have an account? [Sign in to comment](#)

### Assignees

No one assigned

### Labels

No labels

### Type

No type

### Projects

No projects


Milestone

No milestone

Relationships

None yet

Development

 Code with Copilot Agent Mode

▼

No branches or pull requests

Participants

