BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Secure Coding Guidelines for Python

Jan Zádrapa*

Abstract
Many programmers can code efficiently but not securely. This work is focused on the problem of secure coding, specifically on secure coding in Python. This work aims to raise awareness about the issue of secure coding and its use in working and educational environments.
Two main goals are set — first, the guidelines that could be used as educational material and the web application, which is supplementary to the guidelines and serves the purpose of a helpful learning tool for users.
After reading this work, users should understand which coding techniques are secure and effective and which modules and methods they can safely use. In the end, this work could lower the cost of mistakes programmers tend to make.

Keywords: Secure coding — Python programming — Coding guidelines

Supplementary Material: Online guidelines

*xzadra03@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

With the number of cyber-attacks and their price on the rise, the demand for secure coding has also risen. Python is an indivisible part of this problem as the favorite programming language. Many programmers code in Python, but not securely. Python does not have any official, secure coding guidelines and its educational materials on this topic are insufficient.
In most companies, security is often the second thing on their mind after usability [1]. Furthermore, many companies care about security only when time is left. This situation has to be changed. There are many pointers to application insecurity, such as OWASP top ten, which shows that many applications are vulnerable to attack [2].
It is impossible to cover all existing vulnerabilities, and there will be even more vulnerabilities with new technologies coming and evolving. However, programmers should be able to avoid as many of them as possible because damage repair is costly and unnecessary if we can prevent attacks by writing secure code. For example, this latest security discovery shows that code can be executed through comments [3].
This work aims to write secure coding guidelines for Python and create a usable and secure learning tool. It requires only basic knowledge of security topics to understand these guidelines and the learning tool. I know this because the application was tested on people who have only basic knowledge of secure coding. They showed better results in the application's test after reading the guidelines than on the first try before reading the guidelines. These guidelines should be used as an educational tool, and they are suitable for schools and companies. They should be easy to use because, in development, there was an aim on the usability of this product.
The following sections are divided into essential parts when dealing with secure coding. Section 2 of this article defines secure coding and its importance in Information Technology. It also shows existing methods, standards, and learning tools. Section 3 shows existing apps on this topic. Then will be shown secure coding guidelines and implementation of the learning tool (Section 4). The final sections 5 and 6 shows real-life experiments and summary of this work.

2. Existing solutions and standards

Secure coding is a set of practices that applies security considerations to how the software will be coded and encrypted to best defend against cyber-attacks or vul-

nerabilities [4]. Several existing standards and methods can help develop secure software when needed. There are official standards like ISO 27001 [5] and NIST 800-160 [6], and there are community-based and open like OWASP [7] (OWASP Top 10) or CVE [8]. This section introduces these standards and explaining their importance when developing secure software.

2.1 OWASP

The first significant project dealing with secure coding is the Open Web Application Security Project (OWASP). A community-based project founded in 2001 by Mark Curphey. His main goal was to raise awareness about security breaches in applications by identifying the most critical risks. OWASP is a non-profit organization based in the USA [7].

The main goal of OWASP is to inform developers about the most critical security risks in development. For this purpose, OWASP developed several projects, with the most significant one, the OWASP Top 10 [2]. The OWASP Top 10 is used as a measurement mainly for web application developers. In 2021 The OWASP Top 10 looks as follows:

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

My new coding guidelines for Python include a description of every vulnerability from The OWASP Top 10 and its solution. It is considered very important since the enormous public interest in web applications.

2.2 NIST for cybersecurity

NIST develops standards, guidelines, best practices, and others. There are many sections where NIST is creating standards. One of these sections is IT, precisely cybersecurity. Part of NIST is also NICE (National Initiative for Cybersecurity Education). NICE's goal is to educate the public about cybersecurity and train developers. NICE is also aiming at cryptography and cybersecurity measurement [9].

NIST 800-160

NIST 800-160 is called Systems Security Engineering. The main goal of this standard is to raise awareness about security problems when developing is a system. NIST 800-160 suggests strategies how for accomplishing a secure system. It is a standard for developers and management, but it is very comprehensive. It deals with many problems [6].

2.3 CVE

The Common Vulnerabilities and Exposures (CVE) is a list maintained by The Mitre Corporation, and it was launched in September 1999. Since then, almost 173 thousand vulnerabilities have been added to this list. Every vulnerability has its CVE ID, publishing date and updates, type, and score. The score sets the severity of the vulnerability, and its scale is 0 to 10, with ten being the most relevant. There is a special list for Python [10], which consists of 68 vulnerabilities. Some of them are for older versions of Python, but some vulnerabilities can be exploited even in the latest Python versions.

3. Existing tools

This section is focused on existing tools. Many static analysis tools exist, but they are not educational so they will be skipped. There are a few tools worth mentioning. Every tool is described how it works. Of course, there are many other tools and applications for secure development. However, I wanted to point out these three because they are accessible even without booking demos, and the applications are educational, which is the vital point here.

3.1 Codebashing

The first tool that is good to point out is called Codebashing [11]. Codebashing is a tool developed for developers who want to learn about security problems. Checkmarx developed it. The primary purpose of this tool is to teach secure coding techniques in various languages, including Python (Django). The first two lessons are free. For registration, the user has to have a company email. I tested this tool with BUT student email. There is a quiz question at the end of every lesson focused on lesson fundamentals and then the lesson's summary. I think that the main problem with this tool is the paywall after two lessons. Checkmarx likely aims to Codebashing on enterprise usage, and there is a place for pay applications. This attitude is not suitable for the public educational tool.

3.2 Secure Code Warrior

The second educational tool is called Secure Code Warrior [12]. The same name company created this tool. This tool is even more interactive than the Codebashing. There is only a free trial for Python Django, but developers can choose between almost 30 different languages or frameworks. There is Django, Flask, basic Python, and Python API for Python.
The goal is to repair insecure code. The first task for the user is to find a security flaw in the code. Several parts of the code are marked with warning signs, and his goal is to mark the right one. Then when the correct code is marked, the user has to repair the part of the code he has just marked. There are four possible solutions, and he has to choose the right one. There is a theoretical text about the current problem on the left side of the screen. Once this is done, the user can move to the next part.

3.3 Avatao

The difference between Avatao [13] and the tools above is that Avatao is only for eight languages. Fortunately, Python is one of them. This tool is like Codebashing, but it contains more explainable text, which should be more educational. The tool aims more on the standard Python than previous tools. The trial lesson was about input checking, and I appreciate that such a concern could be learned for free.

3.4 Guidelines for other languages

Even if Python does not have official guidelines, some languages already have one. These guidelines can be helpful even when developing in Python. Not in a syntactic way, because every programming language has its unique syntax, but the ways of dealing with some problems are more or less the same. For example, Java has guidelines directly on the Oracle website. It is an enormous file that covers many Java problems [14]. Python developers could want to look into these because both languages are object-oriented, and some problems could have similar solutions. This thesis, Chapter 5, should be used as guidelines for Python.

3.5 Summary

The tools are insufficient. Most of these tools are for web application development. They can help prevent OWASP top 10 vulnerabilities, but they can not teach much about generic Python vulnerabilities such as input validation. This work aims to securely teach the user to code web applications and generic Python code. A learning programmer does not know what he will be coding during his professional life. Python development is not only about OWASP. Also, most of the tools are only code analysis tools. Programmers should know what to do and how to code securely and then look for analysis tools.

4. Guidelines and SeCoPy

Guidelines and learning tools are why this work has even originated. Guidelines are formed in the form of a list of vulnerabilities and its solution. It has not covered all of the vulnerabilities because that is not possible. The goal was to pick out the most common and worst vulnerabilities from my view. Guidelines were divided into three parts chosen because basics are essential for every programmer, and especially in Python, there are things that programmers should handle. The standard library is the core of Python, and there are countless vulnerabilities there. Finally, since web programming is popular nowadays, it has to be included in its part.
The learning tool or SeCoPy (Secure Coding in Python) should be usable and secure. The SeCoPy is vital because guidelines themselves are not enough. For some users, it might be tedious when their only concern is to read the guidelines, and that is all. This learning tool should bring some feedback because users can test their knowledge.
The aim is to make a tool and guidelines that are free for public use, contain many examples, and give the user some feedback. These features should be enough to place SeCoPy among other learning applications.

4.1 Secure coding guidelines

The primary division of guidelines is into three parts. Each part should be a critical topic of secure coding. The first part is dedicated to the basics. Basics consist of these issues:

- Version of Python
- Virtual environment
- Importing modules

These three items are the most important because many vulnerabilities have been already patched in the latest versions. The virtual environment can prevent the spreading of malicious modules or other issues in the project, and the correct module import is crucial.
For example, a vulnerability called typosquatting exists, which uses programmers' carelessness when importing modules. The one typo could have a massive impact on the project.

```
import module
import modulee
```

The second part of the guidelines is focused on the standard library. Standard library means every module available in the library without downloading through PYPi or other Python module downloader. This part is divided into programming techniques and standard library flaws. The first part consists of:

- Input validation
- If-else vs. try-except

And the second part:

- Temporary files
- Pickle module
- Command injection
- Regular expression
- String formatting
- XML
- Random
- Assert
- Zip files
- Command code execution

These issues are the most concerning from my point of view. Some are more about good practice, and some are about using the correct methods—for example, the difference between blacklisting and whitelisting Blacklisting is a technique that creates a blacklist with forbidden values. This technique, in most cases, is not effective nor secure because there are almost infinite inputs that the user can create—on the other hand, whitelisting create a whitelist with allowed input values. One right condition can help secure the user input and is effective. Regular expressions are often used for whitelisting conditions. The last part of the guidelines is focused on web programming. The main focus is on OWASP top 10 2021. In this list, there are the most concerning things about web programming. In all cases, the problem itself and the solution proposal are explained.

```
#blacklisting, trying to catch all mistakes
if len(input_text) == 5:
    if not input_text.isdigit():
        if input_text.startswith("H"):
            if input_text.endswith("o"):
                #still can print Hallo
                print(input_text)
            else:
                print("Wrong ending.")
        else:
            print("Wrong beginning.")
    else:
        print("Error, includes digits.")
else:
    print("Wrong length.")
```

Figure 1. Bad practise using blacklisting.

```
import re
#whitelisting with regex
if re.search("^Hello$", input_text):
    print(argv[1])
else:
    print("Wrong input.")
```

Figure 2. Good practice using whitelisting with regex.

4.2 SeCoPy - The learning tool

The second practical part of this work is developing a learning tool for users. This tool should be usable and also secure. The tool has been called SeCoPy, which means Secure Coding in Python. SeCoPy is developed in Django, a framework for web applications for Python. Front-end was created with the help of Bootstrap for better and faster results. The aim is on quantity because many applications are focused on one topic.

While designing the app, it was necessary to clarify how the workflow would look. The primary use cases would be: wanting to learn, taking the test, and looking at the examples. It was more comfortable to determine the workflow with these use cases defined. There should be an isolated theoretical part where the user learns vulnerabilities and standards because some users could want only to learn. If not after learning, the user should continue on the test, where every chapter of guidelines should be included because the test has to cover every chapter guidelines. It should also have practical questions with code since the goal is to teach the user the best it can, and practical questions are made for this purpose. Many examples included in this application would be friendly for the user if he could download and try the examples himself. These are the main thoughts before the development itself.

The learning phase is about guidelines and theory. The main goal of this part is to teach users about secure coding and already existing methods and, most importantly, to show the guidelines. The guidelines are understandable for the user and include many examples.

The second part of SeCoPy is a test consisting of twenty questions now. There are five questions on every topic because of the consistency. One of the questions is shown in Figure 3. The user should test his knowledge on every chapter equally. Every question has five different answers with one answer right. It includes the answer ”I do not know” because of an implementation problem with radio selection in JavaScript. Every time the submit button is clicked,

one answer must be filled in for desired functionality. After submission, the number of correct answers appears below the button, and the user chooses between retaking the test and showing the correct answers.

The last section is used as a data store for examples

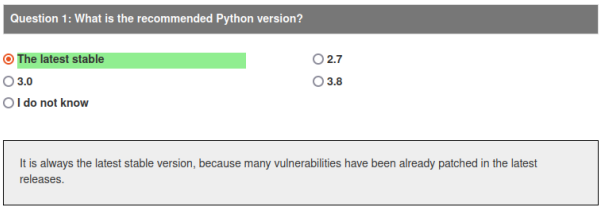

Figure 3. The look of the question.

used in guidelines. Several Python scripts are shown in the guidelines that users can download and test themselves. The SeCoPy is hosted with free hosting on Heroku at the moment [1]. On this website, users can try the application or read the guidelines. This application aims to educate users for free and not only about one topic. There are instructions on what users can do in the application on the home page.

5. Experiments

When testing the guidelines, some experiments were done. Some of them are already in the SeCoPy app, but others are in the making. Another web application was made to test SQL injection [15], which will also be available for users as a playground. In this application, the user could see what SQL query is executed and with the user database also sees what users are picked. Several malicious inputs could lead to unexpected results. Users can see them in help which is placed on the bottom part of the app.

The second experiment worth mentioning is executing a command from comment [3]. This vulnerability is relatively new, and it allows adding executable keywords, such as return and so on, to comments. Precisely to multi-line comments in Python. Here is an example of a malicious comment 4. The problem is that the editor (user) sees the code differently than the interpreter. This flaw can be a concern when downloading an open-source application, where one comment is not noticeable because of the number of lines of code. The RLI character stands for Right-to-Left-Isolation or U+2067, which is invisible and converts the reading order. Changes have been made in Visual Studio Code or GitHub, where users can see these hidden characters now, but this does not apply to all source codes on the internet.

[1] URL: https://secopy.herokuapp.com/

```
'''If it is user in database then login and
return;'''

'''If it is user in database then login and
RLI''';return
```

Figure 4. What user sees vs. what interpreter sees.

6. Conclusions

This work explained the need for secure coding and the existing solutions and materials. The most important is the implementation of the new educational tool SeCoPy and new guidelines for coding in Python, which should ensure that programmers will have learning materials.

This work should emphasise the importance of secure coding, and programmers should be cautious. There are too many vulnerabilities, even in Python, which, as the work suggests, does not have the most significant security concerns as C language. There are no official Python guidelines, and Python developers should consider writing them. This work is only the fundamental contribution to what could be massive official guidelines just as good as Java has. I hope that somebody recognizes the potential and starts working on the first official Python guidelines. It is not likely that I will continue on this project after graduating because it is too much on one person.

Developers should use SeCoPy to code more securely. Students should use SeCoPy for getting knowledge about secure coding. My work is the foundation of what could be the official, secure coding guidelines for Python, as I mentioned in the paragraph above. This work is written when Python 3.10.0 was the latest stable release. It is crucial to keep up even these guidelines up to date. Everyone reading this work should take the final thought: The programming will never be secure.

Acknowledgements

I would like to thank my supervisor, Mgr. Kamil Malinka, Ph.D. for his support, advice, and helpful points while working on this project.

References

[1] Monique Magalhaes. Security vs. usability: Does there have to be a compromise? blogpost (english), Dec 2018. https://techgenix.com/security-vs-usability/.

[2] Inc. OWASP Foundation. OWASP Top Ten. blogpost (english), Mar 2022. https://owasp.org/www-project-top-ten/.

[3] Krebson Security. ‘trojan source’ bug threatens the security of all code. online, 11 2021.

[4] NTT Security AppSec Solutions Inc. Secure Coding. online, 03 2022.

[5] ISO. ISO/IEC 27001. online, 3 2022.

[6] Ron Ross, Michael McEvilley, and Janet Carrier Oren. Systems security engineering. online, 2017.

[7] OWASP Foundation. OWASP. online, 2021.

[8] The MITRE Corporation. CVE Program Mission. online, 2022.

[9] National Institute of Standards and Technology. Cybersecurity. online, 2022.

[10] The MITRE Corporation. Python: Security Vulnerabilities. online, 2022.

[11] Checkmarx. Codebashing. online, 2021.

[12] Secure Code Warrior Limited. Secure code warrior. online, 2021.

[13] Avatao. Avatao. online, 2021.

[14] Oracle. Secure coding guidelines for java SE. online, 09 2020.

[15] OWASP Top 10 team. A03:2021 – Injection. online, 2021.