

CS230-Report

NANDINI CHITALE, RUTH JOHNSON, SHUYANG LIU, XUFAN WANG, CAROLYN YEN,
and HAIKUO YIN

ACM Reference Format:

Nandini Chitale, Ruth Johnson, Shuyang Liu, Xufan Wang, Carolyn Yen, and Haikuo Yin. 2019. CS230-Report. 1, 1 (June 2019), 2 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

3 R's

-Python style

-Databases with vulnerabilities

2 RELATED WORKS

3 METHODOLOGY

Pylint

Bandit

3.1 Dataset

For our data set, we used Safety DB [ref], an online repository maintained by [pyup.io](https://pypi.org/project/safety/) which keeps a list of Python packages with known vulnerabilities.

The data is listed on their Github in an json file, which contains, among other things, the name and version number of the vulnerable packages. We wrote a python script to parse the file (called `insecure_full.json`), retrieve the name and version number, and attempt to download the package through pip download.

When running the script on Windows, out of the 630 packages listed in the json, pip was unable to download 154 of them, either because pip cannot find the package or because none of the available versions have the vulnerability. When repeated on Ubuntu 18.04, pip was unable to download 160 of them. Thus, we will be using the packages downloaded when the script ran on Windows.

Authors' address: Nandini Chitale; Ruth Johnson; Shuyang Liu; Xufan Wang; Carolyn Yen; Haikuo Yin.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

XXXX-XXXX/2019/6-ART \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

4 RESULTS

5 DISCUSSION

6 SECURE PYTHON CODING STANDARDS

As the results have shown, the relationship between PEP8 coding standards and potential security vulnerabilities is not very clear. This is expected. PEP8 is intended for software developers to write more readable code as code is read more often than it is written[1]. However, PEP8[1] is not strong enough for ensuring the security of the program. Therefore, what developers really need, perhaps, is a set of coding standards that prevent common security vulnerabilities. As mentioned in the Related Works section, various coding standards have been proposed for different languages such as Java, C/C++, and Perl[2]. However, as far as we know, there is no such standards for Python yet. Due to the time limitation, we pick five of the most critical security vulnerabilities identified by Bandit that appears most frequently in our data set. In this section, we propose the recommended coding conventions and standards that programmers can follow to avoid the potential attacks due to these vulnerabilities. Some of the rules are similar to those for other languages, some of them are Python specific. We focus on the general usages of Python instead of specific popular Python packages such as Django in this report. Thus, we only consider the usages of the Python standard library here.

6.1 Sanitizing User Inputs

6.2 Do not use assert to enforce constraints

6.3 Do not use wildcard in file system path

6.4 Use of input function

6.5 Serializing and Deserializing with marshal

7 FUTURE WORKS

8 CONCLUSION

9 APPENDIX

REFERENCES

[1] Pep 8 – style guide for python code. <https://www.python.org/dev/peps/pep-0008/>. Accessed: 2019-05-31.

[2] Sei cert coding standards. <https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards>. Accessed: 2019-05-31.