ICAI 2023
12th International Conference on Applied Informatics
Eger, Hungary, 2023

Improper Input Validation Vulnerability Detection by Static Analysis*

Alqaradaghi Midya[ab], **Kozsik Tamás**[c]

[a]ELTE Eötvös Loránd University
alqaradaghi.midya@inf.elte.hu

[b]Northern Technical University
midya.azad@ntu.edu.iq

[c]ELTE Eötvös Loránd University
kto@inf.elte.hu

Abstract

Web applications have become a vital element of our everyday lives. Because they contain valuable and sensitive information, hackers attempt to identify and exploit vulnerabilities to deceive the user, steal data, or destroy the application.

Improper input validation (IIV) [6], also known as unchecked user input, is a type of computer software vulnerability that can lead to security feats. It occurs when a product fails to validate or incorrectly validates input that can affect a program's control flow or data flow. IIV is one of the most prevalent and high-risk vulnerabilities to this day. This vulnerability has been in the CWE Top 25 for the previous four years [9], under the ID CWE20. It is also the root cause of more than half of the Top 10 Web Application Security Risks list [10] such as Cross-site Scripting [7], Command Injections [5], and HTTP Response Splitting [8].

Static analysis techniques are used to find programming errors and code violations, including IIV, without running the programs. Nevertheless, static analysis can be used effectively in the early stages of software development. SpotBugs is an open-source and free static analysis tool [4]. Along with its plug-in for security audits of Java web applications, Find Security Bugs (FSB) [3], they both had a high performance in identifying different programming violations [1]. Moreover, FSB has eight checkers that target IIV vulnerabilities in Java source code.

*The Stipendium Hungaricum scholarship supported this research

Even though IIV is easy to detect and fix, it still occurs frequently in practice [2]. This study is an investigation of IIV vulnerability existence in recent open-source Java web server applications and the effectiveness of existing static analysis techniques in detecting it.

The research questions are:

1. How frequently do recent web-server applications include IIV vulnerability?
2. How effective do recent free and open-source static analyses identifying this vulnerability?
3. How to improve the vulnerability detection rate (Recall)?

To answer the first question, we manually reviewed three web-server applications and checked if they contained IIV vulnerabilities, then documented the number and type of each one. By doing this, we get the total number of IIV in the applications.

To answer the second question, we analyzed the same applications using FSB. We considered recall (also called detection rate) 1 as the basis for the analysis efficiency.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

Where TP: true positive (IIV found by FSB). FN: false negative (number of IIV that FSB has missed). The denominator TP+FN: is (IIV found by manual review).

Table 1 shows the number and types of IIV vulnerabilities found by manual review and FSB in this study's analyzed applications (APP), along with the recall value for each. From the table, we can see that FSB could not detect specific types of IIV vulnerability resulting in recall values of 0.46, 0.74, and 0.63 for the analyzed applications, respectively.

Now we can answer the last question of this study. For improving the vulnerability detection rate, the static analysis should consider detecting more ranges of invalidated inputs, precisely those types that have been missed by the FSB tool; input from buffered readers, and session cookie values.

To conclude, the primary results of this study show that FSB could only identify 37% of the existing IIV vulnerabilities in the analyzed Java applications, meaning it was ineffective in that aspect. Static analysis should consider the *buffered reader* and *session cookies value* input types when looking for IIV vulnerability in Java source code.

References

[1] M. Alqaradaghi, G. Morse, T. Kozsik: *Detecting security vulnerabilities with static analysis – a case study*, Pollack Periodica 17.2 (2021), DOI: 10.1556/606.2021.00454.

Table 1. Number and types of IIV vulnerabilities found by both manual review and FSB tool along with the value of recall for FSB tool when analyzed each application.

APP ID	IIV found by manual review	IIV found by FSB	Recall
1	1 (buffered reader) 4 (query string) 3 (servlet parameter) 5 (session cookie value)	3 (query string) 4* (servlet parameter)	0.46
2	1 (reading a buffered reader) 3 (query string) 12 (servlet parameter) 3 (session cookie value)	9* (query string) 1 (session cookie value) 10 (servlet parameter)	0.74
3	1 (buffered reader) 3 (query string) 4 (servlet parameter)	3 (query string) 2 (servlet parameter)	0.63
Total	40	25	0.63

[2] L. Braz, E. Fregnan, G. Çalikli, A. Bacchelli: *Why Don't Developers Detect Improper Input Validation?'; DROP TABLE Papers;–*, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 499–511.

[3] F. S. Bugs: *An extensible cross-language static code analyzer.* https://pmd.github.io/., Accessed: December 2022.

[4] SpotBugs: *Find Bugs in Java Programs*, https://spotbugs.github.io/, Accessed: December 2022.

[5] The MITRE Corporation: *Command Injection*, https://cwe.mitre.org/data/definitions/78.html, Accessed: December 2022.

[6] The MITRE Corporation: *CWE20 Improper Input Validation*, https://cwe.mitre.org/data/definitions/20.html, Accessed: December 2022.

[7] The MITRE Corporation: *CWE79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*, https://cwe.mitre.org/data/definitions/79.html, Accessed: December 2022.

[8] The MITRE Corporation: *HTTP Response Splitting*, https://owasp.org/www-community/attacks/HTTP_Response_Splitting, Accessed: December 2022.

[9] The MITRE Corporation: *Top 25 Most Dangerous Software Weaknesses*, https://cwe.mitre.org/data/definitions/1387.html, Accessed: December 2022.

[10] The MITRE Corporation: *Top Ten Web Application Security Risks*, https://owasp.org/www-project-top-ten/, Accessed: December 2022.