





Security Testing for Chatbots

Josip Bozic^(✉)  and Franz Wotawa 

Graz University of Technology, Institute for Software Technology, A-8010 Graz,
Austria

{jbozic,wotawa}@ist.tugraz.at

<http://www.ist.tugraz.at>

Abstract. Services like chatbots that provide information to customers in real-time are of increasing importance for the online market. Chatbots offer an intuitive interface to answer user requests in an interactive manner. The inquiries are of wide-range and include information about specific goods and services but also financial issues and personal advices. The notable advantages of these programs are the simplicity of use and speed of the search process. In some cases, chatbots have even surpassed classical web, mobile applications, and social networks. Chatbots might have access to huge amount of data or personal information. Therefore, they might be a valuable target for hackers, and known web application vulnerabilities might be a security issue for chatbots as well. In this paper, we discuss the challenges of security testing for chatbots. We provide an overview about an automated testing approach adapted to chatbots, and first experimental results.

Keywords: Adaptive systems · security testing · chatbots

1 Introduction

Since Joseph Weizenbaum [27] introduced ELIZA, the first computer program that interacts with users in a natural language, in 1966, humanlike communication with a machine has been of growing interest, leading to improvements, e.g., see [14,26] and finally to chatbots, which rely on artificial intelligence for emulating natural conversation with humans. Whereas some chatbots realize conversation using pre-specified patterns, others make use of machine learning techniques [19,23]. These intelligent chatbots provide the user a more personalized conversation by remembering and reusing specific information from previous conversations. Chatbots have also gained a lot of interest from industry. The evolution of these systems over the years has been analyzed and there are predictions about the rise of the chatbot market in the future [5,7].

Like any other technology also chatbots do not come without drawbacks. Despite their intuitive novelty, chatbots are built upon existing technology. They are often integrated into online websites. Therefore, they rely on HTTP(S)

and other existing communication protocols. Smart chatbots are connected to databases, thereby performing SQL queries. Data integrity and privacy, as well as user authentication and authorization must be ensured to the clients, especially by personalized chatbots. If a chatbot fails in this task, data leaks can compromise the user’s privacy and may lead to financial losses. Because of these facts, it is very likely that even chatbots become a target for attackers, where known vulnerabilities and attacks, like cross-site scripting (XSS) and SQL injections (SQLI), can be exploited. Therefore, it is inevitable to cover security issues when testing chatbots as well.

In this paper, we discuss the issue of security testing for chatbots, where we describe an automated approach for the detection of intrinsic software leaks in order to prevent their exploitation. We do not test the chatbots’ performance nor their functionality, e.g. natural language processing, or ask what the underlying machinery should be allowed to do [21]. We solely focus on security testing. The result is an offensive testing approach that targets two very common exploitations, namely XSS and SQLI, which has – to the best of our knowledge – not been considered before in the context of chatbots.

The paper is organized as follows. Section 2 gives an overview about the overall testing approach. Then, Section 3 explains a concrete example and discusses the outcome. Section 4 enumerates related work, whereas the paper is concluded in Section 5.

2 Overview of the approach

When designing chatbots, the primary focus lies on the processing of natural language. There the developers must take into consideration the correct understanding and answering of the user’s inquiries. In addition, the system should be able to handle errors and unexpected inputs appropriately [6]. Existing tools [1, 2, 11] primarily target the system’s functionality but do not guarantee security. The chatbot can fulfil its functional requirements but still remain vulnerable to malicious actions. The still open challenge is to test chatbots regarding their resistance to unintended and malicious user inputs. Figure 1 depicts an overall structure of an online system comprising a chatbot.

In this example, a chatbot is set up online and the communication proceeds accordingly to the standard HTTP(S) protocol. We further assume that the chatbot is connected to a database comprising client-related private information. A smart chatbot would be able to increase the amount of information about a user during communication. Therefore, user authentication must be guaranteed as well as the integrity of all stored information. [9, 15, 17] showed that several web vulnerabilities can be exploited due to security leaks in systems. For example, the vulnerabilities SQLI and XSS can be triggered because of insufficient input sanitization. The consequence can be unauthorized database access or malicious script execution on side of the client, which has to be avoided.

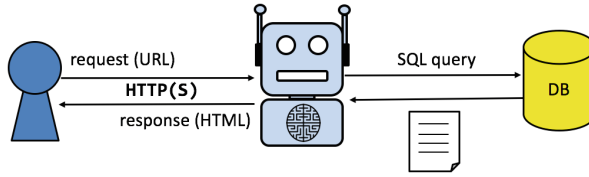


Fig. 1. Communication flow in the chatbot system

For our approach to chatbot security testing, we rely on an adapted execution framework and test oracles for both types of vulnerabilities from previous work [12]. This framework comprises two test sets for XSS and SQLI, respectively. Both test sets consist of a list of individual malicious inputs, called attack vectors. In case of XSS, the list encompasses JavaScript code, and for SQLI, a list of SQL statements is used in the tests. The test inputs are sent sequentially to the chatbot, i.e., the system under test (SUT). The resulting outputs from the chatbot are read and checked against the test oracles, and a test verdict is given back as a result. Figure 2 depicts the overall approach.

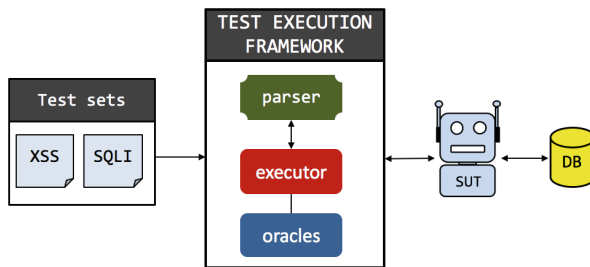


Fig. 2. Security testing approach for chatbots

The framework is implemented in Java and comprises several elements. Both test sets are attached to an executor. Then, every attack vector is put into generated HTTP requests individually and sent against the SUT. A HTML parser [8] reads the corresponding output in search for critical content that is needed by the test oracle. Finally, the testing procedure terminates when both test sets have been exhausted. Section 3 will describe the test scenario in more detail on an example.

3 Case Study

In this case study, we used the described approach for security testing a chatbot. Although several chatbots are developed and used by private companies, some of them are publicly available, e.g., [3, 4]. For this case study, we selected *Program*

O, which is written in PHP and comprises a MySQL database [10], because it perfectly fits the structure provided in Figure 2. *Program O* makes use of conversational patterns that have to be specified in the Artificial Intelligence Markup Language (AIML) [25]. According to given patterns, the chatbot formulates its responses by analyzing the user's provided keywords.

For security testing *Program O* we developed test suites for XSS and SQLI. Scripts for XSS have the following form:

```
<script>alert(document.cookie)</script>">
```

For SQLI, we add SQL queries containing for example the following code:

```
' OR 1=1 #
```

When testing the chatbot, we make use of the provided input field used to communicate with humans. According to the mentioned test oracles from [12], we draw the conclusions for the obtained test verdicts.

When testing with the test suite for XSS, the parsed response from the SUT indicates that the script was not triggered. Unfortunately (for the attacker), critical parts, namely the `<script>` elements, were filtered out from the input string thus preventing its execution. The response HTML contains only a fraction of the original script, e.g. `alert(document.cookie)">`.

We obtained a similar result for SQLI attack related test input where the depicted attack vector is meant to retrieve data from the database. The chatbot's HTTP response body shows evidence that this input has been filtered out as well (by escaping out the apostrophe `'`, which is enough to prevent the execution).

Although, we were not able to successfully trigger a vulnerability for *Program O*, the testing framework at least showed evidence that *Program O* has no trivial security bugs. In addition, we showed that, the overall challenge of security testing of online chatbots can be reduced to general security testing for web applications. In both cases, the same challenges exist, namely how to define attack vectors and how to construct efficient detection mechanisms.

4 Related Work

To the best of our knowledge, there are no papers dealing with security testing of chatbots. There are papers describing methods and tools for testing functionality and usability (e.g. [24]), and also other papers considering testing of AI systems in general [18, 22]. In the general context of security testing there has been some publication dealing with testing against certain attacks like XSS and SQLI.

In [20] the authors present QED, a system that is based on goal-directed model checking for testing against XSS and SQLI. It uses a definition of the vulnerability to be tested and a set of input values for test case generation. QED targets on automated testing of Java web applications. There a model checker is used to generate attack vectors for the SUT via searching for candidates that are likely to detect a vulnerability.

Duchene et al. [15] present a testing tool for XSS that relies on fuzzing and model inference. The underlying method is a black-box fuzzer and makes us of a genetic algorithm with the help of an attack grammar. The work sets focus on the input generation of XSS attack vectors by applying mutation and crossover operators. A fitness function guides the choice of inputs for test case generation, which attack vectors are then executed against web applications.

In our previous work [12], we use visual depictions of attacks against web applications. There we specified attack patterns for XSS and SQLI that guide test execution. The result is an abstract state machine that offers a high degree of configurability and extendibility for black-box security testing purposes.

Other works that cover XSS and SQLI include [16] and [13].

5 Conclusion and Future Work

In this paper, we introduced a first version of a security testing approach to be applied to chatbots. We claim that the topic covers a challenge of growing interest and importance. This is due to growing interest of chatbots from industry (see [7]). Most interestingly, the scientific literature lacks solutions for the challenge of testing chatbots for vulnerabilities. In this paper, we briefly introduced a testing framework for security testing chatbots and discuss first results we obtained using an available chatbot implementation.

Although, we were not able to trigger vulnerabilities, we could show that the framework fits well its purpose. In addition, the tests raise evidence that the used chatbot is resistant to some common attack vectors. It is worth noting that the current test execution framework is not limited to XSS and SQLI attacks. In the future, we will extend testing chatbots against other vulnerabilities [9]. In addition, we want to further investigate on automated test case generation for security testing of chatbots.

Acknowledgments. The research presented in the paper has been funded in part by the Cooperation Programme Interreg V-A Slovenia-Austria under the project AS-IT-IC (Austrian-Slovenian Intelligent Tourist Information Center).

References

1. Botium - new generation testing. <http://www.botium.at>, accessed: 2018-05-07
2. BotMan - The PHP messaging and chatbot library. <https://botman.io>, accessed: 2018-05-20
3. BotMill.io - We Mill Bots and Create Bot Milling Tools! <http://www.botmill.io>, accessed: 2018-05-22
4. CharlieBot. <https://sourceforge.net/projects/charliebot/>, accessed: 2018-05-22
5. Chatbot Market Size And Share Analysis, Industry Report, 2014-2025. <https://www.grandviewresearch.com/industry-analysis/chatbot-market>, accessed: 2018-05-07
6. Chatbottest. <http://chatbottest.com>, accessed: 2018-05-07

7. Gartner Top Strategic Predictions for 2018 and Beyond. <https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond/>, accessed: 2018-05-07
8. jsoup: Java HTML Parser. <https://jsoup.org/>, accessed: 2018-02-02
9. OWASP Top Ten Project. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, accessed: 2018-01-31
10. Program O AI Chatbot - The Friendly Open Source PHP, MySQL, AIML Chatbot. <https://www.program-o.com>, accessed: 2018-02-04
11. QMetry BOT Tester. <http://www.qmetry.com/qmetry-bot-tester/>, accessed: 2018-05-07
12. Bozic, J., Wotawa, F.: Security Testing Based on Attack Patterns. In: Proceedings of the 5th International Workshop on Security Testing (SECTEST'14) (2014)
13. Clarke, J., Fowler, K., Oftedal, E., Alvarez, R.M., Hartley, D., Kornbrust, A., O'Leary-Steele, G., Revelli, A., Siddharth, S., Slaviero, M.: SQL Injection Attacks and Defense, 2nd edn. Syngress, (2012)
14. Colby, K.: Artificial Paranoia: A Computer Simulation of Paranoid Process. Pergamon Press, New York (1975)
15. Duchene, F., Rawat, S., Richier, J.L., Groz, R.: KameleonFuzz: Evolutionary Fuzzing for Black-Box XSS Detection. In: CODASPY. pp. 37-48. ACM (2014)
16. Fogie, S., Grossman, J., Hansen, R., Rager, A., Petkov, P.D.: XSS Attacks: Cross Site Scripting Exploits and Defense. Syngress, (2007)
17. Halfond, W.G.J., Viegas, J., Orso, A.: A Classification of SQL Injection Attacks and Countermeasures. In: Proceedings of the IEEE International Symposium on Secure Software Engineering. Arlington, VA, USA (2006)
18. Liu, G., Liu, Q., Zhang, W.: Model-Based Testing and Validation on Artificial Intelligence Systems. In: Second International Multisymposium on Computer and Computational Sciences (2007)
19. Lowe, R., Noseworthy, M., Serban, I.V., Angelard-Gontier, N., Bengio, Y., Pineau, J.: Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In: Proceedings of the 5th International Conference on Learning Representations (ICLR) Workshop. Toulon, France (2017)
20. Martin, M., Lam, M.S.: Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking. In: 17th USENIX Security Symposium (2008)
21. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 4, pp. 463-502. Edinburgh University Press (1969), reprinted in McC90
22. Rushby, J.: Quality Measures and Assurance for AI Software. In: NASA Contract Report 4187, Washington DC (1988)
23. Shawar, B.A., Atwell, E.: Using corpora in machine-learning chatbot systems. In: International Journal of Corpus Linguistics, vol. 10 (2005)
24. Vasconcelos, M., Candello, H., Pinhanez, C., dos Santos, T.: Bottester: Testing Conversational Systems with Simulated Users. In: IHC 2017: Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems (2017)
25. Wallace, R.S.: The Elements of AIML Style. In: ALICE A.I. Foundation (2003)
26. Wallace, R.S.: The Anatomy of A.L.I.C.E. In: ALICE A.I. Foundation (2004)
27. Weizenbaum, J.: ELIZA-A Computer Program For the Study of Natural Language Communication Between Man and Machine. In: Communications of the ACM Volume 9, Number 1 (January 1966) (1966)