Автоматизація фаззінгу

Войтович О.П.[1], Шашков Р.В.[2]

[1]К.т.н доц. кафедри захисту інформації, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, voytovych.op@gmail.com
[2]Студент, кафедра захисту інформації, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, shashkov1992@gmail.com

***Анотація* — Зроблено огляд методів автоматизованого фаззінгу. Описані різні методи проведення фаззінгу відповідно до етапів проведення дослідження. Запропоновано використання мутаційного методу для формування тестової послідовності.**

Ключові слова: фаззінг, метод проведення фаззінгу,автоматизований фаззінг.

Fuzzing automation

Voytovych O.P.[1], Shashkov R.V.[2]

[1]PhD Assoc. Information Protection Department, Vinnytsia National Technical University
Khmelnytsky shosse 95, Vinnitsa, Ukraine, voytovych.op@gmail.com
[2] Student, Information Protection Department, Vinnytsia National Technical University
Khmelnytsky shosse 95, Vinnitsa, Ukraine, shashkov1992@gmail.com

Abstract - The automated fuzzing methods are researched. Different fuzzing methods during research stages are described. The mutation method for test pattern is proposed.

Keywords: fazzing, method of fuzzing, automated fuzzing.

I. Вступ

Все більше користувачів мережі Інтернет користуються веб-додатками. Це можуть бути звичайні веб-додатки; додатки з важливою комерційною або персональною інформацією, що підлягає захисту; різні платіжні системи, де ризик втрати інформації може оцінюватися в значні суми; додатки з підвищеними вимогами до цілісності; а також популярні і широко використовувані зараз соціальні мережі.

Для того щоб запобігти втрати конфіденційної інформації через веб-додатки потрібно проводити їх тестування [1].

Для тестування веб-додатків використовують автоматизований та ручний фаззінг. Існує багато методів проведення автоматизованого фаззінгу веб-додатків [2]. Перед відправкою даних в додаток, який піддається тестуванню, відбувається їх формування. Саме від правильності сформованих даних залежить ефективність проведення автоматизованого фаззінгу [3]. Дані формуються за певними методами і базуються на генерації псевдовипадкових послідовностей, спеціально сформованих послідовностей, мутованих даних або ж даних, які формуються за допомогою певного алгоритму [4].

Основною проблемою автоматизованого фаззінгу є коректність сформованих даних перед передачею їх у веб-додаток, який тестується, а також правильність аналізу поведінки додатку для виявлення вразливостей.

II. Аналіз методів проведення фаззінгу

Автоматизоване тестування веб-додатку поділяється на декілька етапів, кожен з яких відповідає за деякі функції. Від правильності виконання кожного етапу залежить успіх тестування.

На рис. 1 показано узагальнену схему процесу автоматизації тестування веб-додатку.

Рисунок 1 – Схема процесу автоматизованого фаззінгу

На схемі виділено ряд етапів - формування даних перед відправкою, надсилання даних, отримання результатів, прийняття рішення автоматизованим засобом.

Етап надсилання даних може використовувати такі методи [3,4] як: послідовне опитування портів, вибіркове опитування портів, опитування зі зворотнім зв'язком. На даному етапі відбувається надсилання даних до об'єкту дослідження.

Після того, як дані було надіслано відбувається етап отримання результатів. Отримання результатів може відбуватися [5] від досліджуваного об'єкту, від операційної системи, від веб-серверу, мережевого вузла, між мережевого екрану, також воно може бути взагалі відсутнє.

Після отримання результату виконується етап прийняття рішення на якому аналізуються отримані дані. Прийняття рішення може відбуватися [6]: людиною, логістичне (сигнатурне, статистичне), нечітке (нечітка логіка, нейронні мережі).

Перед відправкою даних відбувається їх формування. Від даного етапу залежить подальша ефективність проведення тестування веб-додатку.

Етап формування даних базується на трьох методах[4]:

1) Випадковий: перша і сама традиційна стратегія для створення вхідних даних для фаззінга є використання абсолютно випадкових даних, без будь-якого інтелекту або спеціальних знань про додатки. Досить велика кількість вразливостей була виявлена, завдяки цій методиці [6, 7, 8, 9,10].

Хоча ця методика вирізняється своєю швидкістю, низькою вартістю і відносно простою у реалізації, але вона має і ряд недоліків. Як правило,таким методом можна тільки поверхнево протестувати додаток [11]. Недоліком методу є те, що він менш ефективний, коли використовуються контрольні суми [12].

2) Мутаційний. При мутаційному методі міститься колекція коректно сформованих вхідних даних. Спотворюючи в таких даних випадковий байт або рядок, отримується черговий тестовий набір. Даний підхід значно ефективніше попереднього. Крім того, створення мутування даних не вимагає значних часових витрат. Проте, найімовірніше, що програма, яка проходить тестування буде відкидати велику частину вхідних даних, яка не відповідає специфікації. Також, покриття коду в даному випадку безпосередньо залежить від того, наскільки добре сформована база коректних даних.

3) Модельний. Найбільш просунутий метод формування даних [12,13,14]. У даному випадку на основі специфікації досліджуваного веб-додатку створюється граматика, в якій вказані змінні, статичні дані, а також динамічно обчислювані величини (наприклад, контрольні суми). Потім на її основі формуються тестові набори. Ефективність даного методу тим вище, чим точніше описана граматика, крім того, вона зазвичай вище в порівнянні з ефективністю мутуючого тестування. До мінусів підходу можна віднести підвищену складність розробки.

Серед трьох методів мутаційний метод має такі переваги, як простота та можливість використання на різному програмному забезпеченні. Недоліком методу є потреба в достовірних матеріалах, щоб досягти максимального охоплення в тестуванні.

Отже, після аналізу ряду переваг та недоліків різних методів формування даних можна зробити висновок, що перспективним методом формування даних є мутаційний метод.

III. ВИСНОВКИ

Отже, було проведено аналіз існуючих методів проведення фаззінгу веб-додатків. Проаналізовано етапи проведення тестування. При аналізі методів формування даних було виявлено, що найкращим із методів є мутаційний, який має ряд переваг серед інших методів.

В подальшому буде проведено розробку нового методу формування даних перед відправкою в основі, якого буде лежати мутаційний метод, що дозволить покращити його та досягти кращих результатів у тестуванні веб-додатків на вразливості.

[1] R. Binder. Testing Object-Oriented Systems. Addison Wesley, Reading, MA, 2000.

[2] Di Lucca GA, Fasolino AR, Faralli F, De Carlini U (2002) Testing Web Applications. In: Proceedings of International Conference on Software Maintenance. IEEE Computer Society Press: Los Alamitos, CA, pp 310–319

[3] Саттон Майкл. Fuzzing. Исследование уязвимостей методом грубой силы./ Майкл Саттон – К.:Символ-плюс, 2010 – 560 с. ISBN: 978-5-93286-147-9

[4] Sofia Bekrar, Chaouki Bekrar, Roland Groz, Laurent Mounier «A Taint Based Approach for Smart Fuzzing». IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012.

[5] G. Zhao, W. Zheng, J. Zhao, and H. Chen, “An heuristic method for web-service program security testing,” in ChinaGrid Annual Conference, 2009. ChinaGrid ’09. Fourth, 2009, pp. 139 –144.

[6] B. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of unix utilities,” Communications of the ACM, vol. 33, no.12, pp. 32–44, 1990.

[7] B. Miller, D. Koski, C. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl, “Fuzz revisited: A re-examination of the reliability of UNIX utilities and services”. Citeseer, 1995.

[8] J. Forrester and B. Miller, “An empirical study of the robustness of windows nt applications using random testing,” in Proceedings of the 4th conference on USENIX Windows Systems Symposium-Volume 4, 2000, pp. 6–6.

[9] N. Kropp, P. Koopman, and D. Siewiorek, “Automated robustness testing of off-the-shelf software components,” in Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on. IEEE, 1998, pp. 230–239.

[10] B. Miller, G. Cooksey, and F. Moore, “An empirical study of the robustness of macos applications using random testing,” in Proceedings of the 1st International workshop on Random testing. ACM, 2006, pp. 46–54.

[11] T. Wang, T. Wei, G. Gu, and W. Zou, “Taintscope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection,” in Security and Privacy (SP), 2010 IEEE Symposium on, May 2010, pp. 497 –512.

[12] M. Vuagnoux. Autodafé: an Act of Software Torture. In 22th Chaos Communication Congress, Berlin, Germany, 2005. [Електроний ресурс] Режим доступу: http://autodafe.sourceforge.net/docs/autodafe.pdf.

[13] Peach Fuzzing Platform [Електронний ресурс]. Режим доступу: http://peachfuzzer.com/.

[14] Sulley [Електронний ресурс]. Режим доступу: http://code.google.com/p/sulley/.