УДК 004.056.53
DOI: 10.31673/2409-7292.2020.035499

Горюк Н.В.

ЗАСОБИ ІНТЕГРАЦІЇ ТЕХНОЛОГІЇ СТАТИЧНОГО АНАЛІЗУ БЕЗПЕКИ ВИХІДНОГО КОДУ У СЕРЕДОВИЩЕ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті досліджуються методи автоматизації та засоби інтеграції технології статичного аналізу безпеки вихідного коду. Досліджено процес аналізу безпеки програмного забезпечення, який реалізується технологією статичного аналізу вихідного коду, та запропоновані методи вирішення проблеми автоматизації та інтеграції технології у середовище розробки вихідного коду. Встановлено перспективний напрямок подальшого розвитку технології статичного аналізу вихідного коду.

Ключові слова: статичний аналіз вихідного коду, SAST, безпека програмного забезпечення.

Вступ. На сьогоднішній при розробці інформаційних систем актуальним є прискорення темпів розробки нового програмного забезпечення, його компонентів та версій. Як результат – збільшення об’єму розроблюваного коду та прискорення випуску нових версій продуктів створює нові вимоги до вже існуючих засобів забезпечення безпеки вихідного коду. Разом з тим набуває актуальності й питання автоматизації та інтеграції засобів забезпечення безпеки вихідного коду у середовище розробки задля скорочення часу, що витрачається на мануальне виконання дій зі статичного аналізу коду, та заходів забезпечення його безпеки.

Актуальність проблеми. Технологія статичного аналізу вихідного коду (Static application security testing, SAST) використовується задля аналізу вихідного коду на предмет знаходження у ньому слабких місць, що потенційно можуть призвести до вразливостей та можливості компрометації розроблюваної інформаційної системи. Так, звіт компанії Microsoft [1] щодо показує, що більшість вразливостей, що були розглянуті у дослідженні, експлуатувалися вже після того, як вийшло оновлення, що виправляє ці вразливості. Незважаючи на те, що виробники постійно випускають патчі щодо усунення вразливостей, кількість експлуатованих вразливостей до випуску виправлень збільшується. Це визначає актуальність проблеми застосування технології статичного аналізу вихідного коду у процесі розробки програмного забезпечення (ПЗ).

Розробники ПЗ впроваджують підхід до проведення тестування розроблюваного ПЗ, який полягає у проведенні різного роду тестувань якомога раніше. Даний підхід зазвичай впроваджується разом з практиками безперервної інтеграції та безперервної доставки (Continuous Integration/Continuous Delivery, або CI/CD) (рис. 1), зміст яких полягає у використанні засобів автоматизації процесу розробки, засобів інтеграції технологій та інструментів, що використовуються при розробці. Рішення щодо аналізу безпеки вихідного коду мають відповідати вимогам щодо проведення статичного аналізу коду та забезпечувати необхідний рівень автоматизації своєї діяльності.

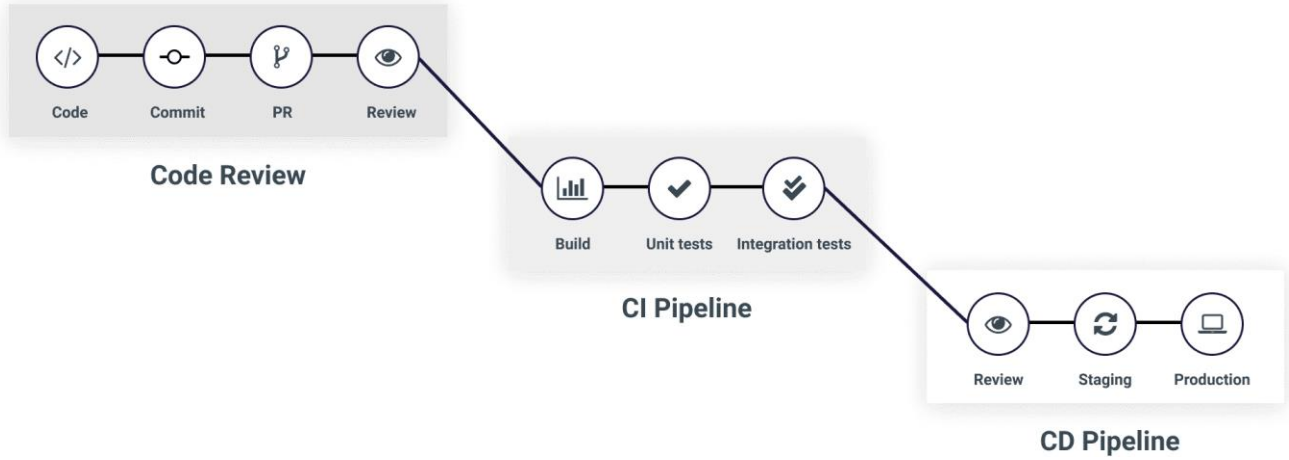

Рис. 1. Етапи життєвого циклу розробки ПЗ відповідно до підходу CI/CD [3]

Причиною для впровадження підходу являється статистика витрат на виправлення вразливостей, які залежить від того, на якому етапі команда розробки виправляє виявлену вразливість. Згідно дослідження “Integrating Software Assurance into the Software Development Life Cycle (SDLC)” [2], виправлення дефектів, що відбувається на стадії тестування ПЗ, коштуватиме у 15 разів більше, ніж на стадії проектування. Дефекти, які були виявлені після релізу версії ПЗ коштуватимуть у 100 разів більше (рис. 2).

Основний матеріал дослідження. Статичний аналіз безпеки вихідного коду реалізується на етапі написання коду, або вже після того, як код написано. Існує декілька підходів до сканування статичного коду – сканування частини вихідного коду без компіляції проекту та перекладу його у бінарний код, а також повноцінне сканування всієї кодової бази з компіляцією коду.

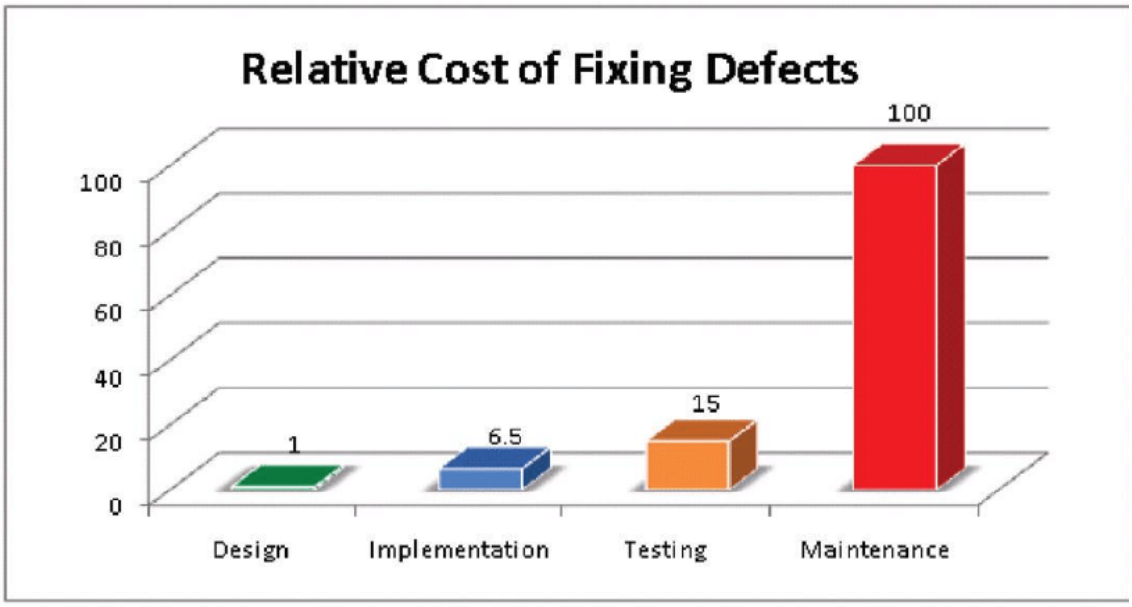

Рис. 2. Відносна вартість виправлення дефектів програмного коду забезпечення [2]

Реалізація технології статичного аналізу для кожного підприємства може різнитися, але, зазвичай, виглядає наступним чином:

вихідний код, що необхідно аналізувати на предмет наявності недоліків подається на вхід до сканеру вихідного коду;

вихідний код аналізується згідно попередньо заданих правил та рамок аналізу;

результати аналізу передаються особам, відповідним за аудит коду задля вилучення хибнопозитивних спрацьовувань та проведення пріоритезації виявлених вразливостей;

аудитори передають результати аудиту для виправлення у виді технічних завдань розробникам ПЗ.

Автоматизації піддаються чотири проміжні етапи процесу аналізу вихідного коду: етап ініціації аналізу вихідного коду, етап аналізу вихідного коду, етап аудиту результатів аналізу безпеки вихідного коду та етап передачі результатів аудиту розробникам.

Перший етап полягає у ініціації процесу аналізу вихідного коду та передачі кодової бази до сканеру. Процес аналізу вихідного коду може бути викликано іншими інструментами середовища розробки через використання прикладних програмних інтерфейсів (Application Programming Interface, API), або через командну строку у тому випадку, якщо інструменти середовища розробки дозволяють використання скриптових мов. Автоматизувати даний етап можуть сучасні системи контролю змін у вихідному коді (Source Control Management, SCM), які використовуються розробниками для контролю за змінами у вихідному коді. Так, наприклад, популярна система контролю змін Git має систему хуків [4], які надають можливість запуску довільних скриптів у разі виникнення певних подій. Наприклад, у

момент запису нової версії коду до кодової бази. Ці ж скріпти можуть й ініціалізувати процес аналізу вихідного коду.

На другому етапі здійснюється сканування вихідного коду на предмет наявності у ньому вразливостей. Сканування здійснюється враховуючи попередньо задані правила, які: визначають рамки сканування коду, спектр шуканих вразливостей, ресурси, що виділяє система на процес сканування та визначають дії, які необхідно здійснити з результатами сканування. Даний етап може бути автоматизовано за допомогою інструментів композиційного аналізу вихідного коду, що визначають мову програмування, зовнішні залежності та їх версійність, та особливості компіляції проекту. На основі результатів композиційного аналізу формується набір команд та правил, що будуть використані у подальшому автоматичному скануванні проекту. Прикладом такого інструменту являється Fortify Scan Wizard [5], який реалізує наведений функціонал та надає можливість автоматично надсилати результати у Центр контролю безпеки ПЗ.

Етап аудиту результатів автоматизованого аналізу проводиться спеціалістами з кібернетичної безпеки, що повинні бути знайомі з використаними у вихідному коді язиками програмування, та архітектурою розроблюваного ПЗ. Під час аудиту спеціаліст переглядає результати аналізу та виключає з них неактуальні та хибнопозитивні спрацьовування сканеру. Автоматизація на цьому етапі досягається за допомогою автоматичної передачі результатів аналізу, класифікації виявлених слабких місць у коді, та надання контекстної інформації щодо вразливостей (наприклад, вказання структури та послідовності функцій, що призвели до виявлення цієї вразливості).

Етап виправлення вразливостей починається з отримання розробниками технічного завдання на виправлення хибних місць у коді. Такі завдання, зазвичай, надаються через систему відслідковування помилок (bug tracking system). Далі, після виправлення хибних місць у коді, процес аналізу вихідного коду повторюється. Якщо нових або старих вразливостей не виявлено, то життєвий цикл розроблюваного ПЗ може переходити на новий етап.

Розглянемо засоби інтеграції технології статичного аналізу безпеки вихідного коду у середовище розробки програмного забезпечення.

В контексті даного дослідження інтеграція визначається як процес поєднання технологій аналізу безпеки вихідного коду з засобами та технологіями розробки коду. Інтеграція надає можливість автоматизувати передачу результатів роботи між технологіями, що дає змогу впровадити нові можливості розробникам програмного забезпечення та спеціалістам з кібербезпеки. Прикладом таких можливостей є інструменти поточного аналізу вихідного коду, що використовуються розробниками під час написання коду, та аналізують написаний код на предмет наявності в них вразливостей.

Технології та інструменти, що використовуються у процесі розробки програмного забезпечення, та до яких можуть бути застосовані засоби інтеграції наступні:

інтегроване середовище розробки (Integrated Development Environment, IDE) – основний інструмент розробника, що являється комплексним програмним рішенням, та складається з редактору вихідного коду і інструментів для автоматизації побудови і відлагодження коду;

система контролю змін у коді/версіях (Version Control System, VCS) – необхідний інструмент для контролю змін у коді та його версіях, що використовується при розробці майже в усіх комплексних проектах. Приклад такої системи – рішення git [6];

інструменти автоматизації побудови коду – даний тип інструментів використовується при розробці для здійснення автоматизації задач побудови проектів та здійснення безперервної інтеграції. Прикладом такого інструменту є рішення Jenkins [7].

Серед засобів для інтеграції з описаними інструментами можна визначити наступні:

програмні плагіни – динамічно підключаються до основної програми та розширюють її функціонал і можливості;

програмні прикладні інтерфейси (Application Programming Interface, API) – забезпечують взаємодію між компонентами програми та різними інформаційними системами за допомогою набору визначених програмних методів та команд;

оркестрація – процес поєднання декількох інформаційних систем разом за допомогою скриптових язиків та можливостей операційної системи.

Інтеграція з IDE виконується за допомогою плагінів, які при підключенні до середовища розробки поширюють її функціонал. Серед функцій, які ці плагіни можуть надати стосовно технології статичного аналізу вихідного коду можуть бути як засоби графічного відображення результатів сканування, так і засоби автоматизації процесу статичного аналізу вихідного коду, що дозволяють віддалено ініціалізувати цей процес. Прикладом такого плагіну може бути Fortify Security Assistant для середовища розробки Visual Studio [8], серед можливостей якого є отримання та відображення результатів аналізу, інтерактивної взаємодії з результатами та центром вразливостей, а також автоматизації сканування та засоби поточного аналізу вихідного коду.

Системи контролю за змінами у коді грають важливу роль у розробці програмного забезпечення. Контроль за змінами у коді надає можливість слідкувати за станом його версій, у тому числі й за станом їх безпеки. Інтеграція з такою системою може відбуватися через засоби оркестрації самої системи контролю за змінами, або за допомогою можливостей операційної системи чи скриптових мов. Прикладом такої автоматизації може бути система хуків у Git.

Наведений нижче (рис. 3) хук автоматично викликає довільний скрипт, що ініціює процес статичного сканування версії проекту після того, як вона потрапила до системи контролю за змінами у коді.

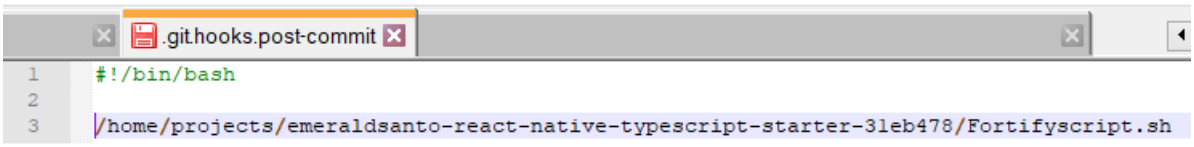

Рис. 3 Скріншот робочого середовища git з post-commit хуком

Інструменти автоматизації побудови коду інтегруються з рішеннями статичного аналізу за допомогою плагінів, що при певних подіях можуть автоматизувати ініціалізацію статичного сканування коду. Відбуватися це може, наприклад, у вигляді однієї з задач при побудові коду проекту, що автоматично викликається під час побудови. Прикладом такого плагіну є плагін Fortify до інструменту автоматизації побудови коду Jenkins [9].

Отже, усі позначені інструменти та технології, що використовуються у середовищі розробки програмного забезпечення, мають можливість проведення інтеграції з рішеннями зі статичного аналізу безпеки вихідного коду. Такі методи інтеграції не лише дозволяють автоматизувати процес аналізу вихідного коду, а й розширити та додати новий функціонал – наприклад, можливість поточного аналізу вихідного коду під час його написання.

Окрім зазначених проблем, розробка засобів інтеграції технології статичного сканування безпеки вихідного коду стала причиною розвитку інших технологій забезпечення безпеки ПЗ, а саме – інтерактивним сканерам безпеки програмного забезпечення (Interactive Application Security Testing, IAST) [10]. Вони широко використовують можливості технології статичного та динамічного аналізу коду, та надають результати аналізу прямо під час його виконання.

Висновки

Розглянуті методи автоматизації процесу виявлення вразливостей вихідного коду, а також засоби інтеграції технології статичного аналізу вихідного коду у процес розробки ПЗ, дозволяють прискорити процес пошуку вразливостей не порушуючи технологічний процес розробки програмного забезпечення за допомогою “безшовної” інтеграції.

Знання методів та засобів інтеграції технології аналізу безпеки вихідного коду, їх розробка та впровадження у життєвий цикл ПЗ є необхідною умовою для вдосконалення процесу розробки безпечного коду, що згодом вплине на стан кібербезпеки корпоративних інформаційних систем, для яких його було задіяно.

Перелік посилань

1. Software Vulnerability Exploitation Trends report, 2013 [Електронний ресурс] - Режим доступу: https://download.microsoft.com/download/7/2/b/72b5de91-04f4-42f4-a587-9d08c55e0734/microsoft_security_intelligence_report_volume_16_english.pdf

2. Integrating Software Assurance into the Software Development Life Cycle (SDLC) [Електронний ресурс] - Режим доступу: https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC

3. Five Ways to Shorten Your Continuous Delivery Cycle [Електронний ресурс] – Режим доступу: https://walrus.ai/blog/2019/11/five-ways-shorten-ci-cd-cycle/

4. Pro Git (український переклад), Configuring Git, Hooks [Електронний ресурс] - Режим доступу: https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

5. Preparing to use the Fortify Scan Wizard [Електронний ресурс] - Режим доступу: https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools/2010/SCA_Help_20.1.2/index.htm#ScanWizard/PrepUseScanWizard.htm

6. Git [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Git

7. Jenkins [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Jenkins

8. Fortify Security Assistant for Visual Studio [Електронний ресурс] - Режим доступу: https://marketplace.visualstudio.com/items?itemName=fortifyvsts.fortify-security-assistant-visual-studio

9. Official Jenkins plugin for Fortify Static Code Analyzer [Електронний ресурс] - Режим доступу: https://plugins.jenkins.io/fortify/.

10. “What is IAST?” - [Електронний ресурс] - Режим доступу: https://www.synopsys.com/glossary/what-is-iast.html.

Надійшла: 08.08.2020 Рецензент: д.т.н., професор Кожухівський А.Д.

.