Столеру Петро Дмитрович, магістр

Чернівецький національний університет імені Юрія Федьковича, м.Чернівці

**ВИЗНАЧЕННЯ ВРАЗЛИВОСТЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ВСІХ СТАДІЯХ РОЗРОБКИ**

У сучасному світі, коли комп'ютерна техніка є необхідною складовою багатьох галузей, у яких безпека даних та захист інформації має першочергове значення. Розробка програмного забезпечення (ПЗ) – не виняток, вже відомо, що більшість вразливостей у ПЗ впроваджуються вже на етапі розробки. Тому виявлення вразливостей на всіх стадіях життєвого циклу програмного забезпечення є надзвичайно важливою задачею, яка допомагає попередити можливі кібератаки та захистити інформацію від несанкціонованого доступу. До того ж згідно з даними досліджень, виправлення вразливостей на етапі розробки ПЗ суттєво скорочують витрати, які б знадобилися на усунення цих недоліків вже після випуску продукту та його впровадження. Тому ця тема є актуальною та важливою для сучасного інформаційного суспільства.

Мета дослідження полягає у вивченні методів виявлення потенційних вразливостей та розробці єдиного рішення для сканування ПЗ на різних етапах його розробки.

Замість цього я би додала

У роботі вирішуються такі завдання:

1. Проаналізувати предметну область застосувань і розглянути програми- аналоги.
2. Дослідити типові вразливості ПЗ та найкращі методи для їх виявлення.
3. Дослідити основні методи виявлення вразливостей на всіх етапах розробки ПЗ, та можливості інтеграції сканеру вразливостей в CI/CD. CI (Continuous Integration) — це неперервна інтеграція, а CD (Continuous Delivery) — неперервна доставка.
4. Розробити застосунок командного рядка для виявлення вразливостей за допомогою 4 типів сканування (SCA, SAST, DAST, API), аналізу всіх бібліотечних файлів та файлів вихідного коду, з можливістю гнучкого налаштування політик безпеки.
5. Надати можливість сканування проектів з систем контролю версій, сканування різних гілок, створення автоматичного pull request з оновленням вразливої бібліотеки та рекомендації щодо усунення, а також сканування контейнерів і докер образів.

Наукова новизнароботи полягає у створені єдиного застосунку командного рядка для виявлення вразливостей з адаптивним налаштуванням політик безпеки для сканування програмних проектів з метою раннього виявлення вразливостей ПЗ в процесі безперервної інтеграції та безперервної доставки в виробниче середовище експлуатації, та підтримка сканування λ-функцій.

Вразливість програмного забезпечення – це недоліки та слабкі місця, якими можуть скористатися зловмисники для здійснення атак. Вона може бути наслідком помилок у коді програми, неправильного використання сторонніх бібліотек або бути спричинена відсутністю необхідних заходів безпеки. Також це можливість використання програми для здійснення несанкціонованої дії, що загрожують безпеці користувача або системи. Вразливість може бути використана для вторгнення у систему, крадіжки конфіденційних даних, розповсюдження вірусів та інших шкідливих програм, а також для виконання інших дій, які можуть завдати шкоди системі.

На кожному з етапів розробки ПЗ від аналізу вимог і планування до підтримки програми важливо захищати свій продукт від різного виду вразливостей. На перших етапах, як-от аналіз вимог та проектування достатньо лише проводити аналіз потенційних загроз, а також проектувати систему з урахуванням заходів безпеки. На ранніх стадіях розробки необхідно тестувати створений код, а також залежності, які використовуються застосунками, для виявлення вразливостей, властивих саме конкретному продукту. Після того, як продукт стає доступний для користувачів, необхідно також підтримувати його безпеку, здійснюючи тестування за методами «білої» та «чорної» скриньок.

Згідно зі статистикою 90% випадків порушення безпеки відбувається через використання зловмисниками відомих програмних помилок [1]. У звіті компанії IBM зазначено, що виявлення дефектів після випуску продукту в 100 разів дорожче за їх розкриття на етапі проектування. OWASP (Open Web Application Security Project) – це некомерційна організація, метою якої є поліпшення безпеки програмного забезпечення [2]. Організація розробляє відкриті стандарти безпеки та методології, які можуть використовуватися в процесі розробки програмного забезпечення, а також збирає та публікує інформацію про вразливості програмного забезпечення та методи їх усунення. Наразі це галузевий стандарт, якого слід дотримуватися при розробці будь-якого ПЗ. OWASP Top 10 містить перелік найбільш критичних ризиків безпеки веб-додатків [3].

Одним із методів виявлення недоліків є сканування вразливостей, яке полягає у використанні багато різних підходів, зокрема тестування на проникнення, аналіз відповідей сервера та перевірка різноманітних параметрів, як-от порти, сертифікати тощо. Інший метод виявлення вразливостей ґрунтується на аналізі вразливостей. Цей метод полягає у вивченні вразливостей та використанні цієї інформації для виявлення можливих прогалин у безпеці. Аналіз вразливостей може бути ефективним, оскільки дає змогу виявити більш складні вразливості та зрозуміти, як вони можуть бути використані для зламування системи. Третій методом виявлення вразливостей – це тестування на проникнення (pennetration testing). Цей метод полягає у використанні спеціальних інструментів для емуляції атак і спроб зламування системи з метою виявлення вразливостей.

У роботі було зроблено акцент саме на виявленні вразливостей, тому розглянемо найпопулярніші методи виявлення вразливостей на всіх етапах розробки ПЗ. Аналіз композиції складу програмного забезпечення (Software Composition Analysis, SCA) – це методи виявлення помилок у залежностях, використаних при розробці сторонніх бібліотек [4]. Статичне тестування безпеки додатків (Static Application Security Testing, SAST) – це тип тестування безпеки додатків, який вивчає вихідний код програми для виявлення потенційних вразливих місць у безпеці проекту. Цей метод перевіряє кодову базу на наявність поширених помилок кодування, до яких належать SQL-ін’єкції або міжсайтовий скриптинг (XSS), а також інші вразливостей, якими можуть скористатися зловмисники [5]. Сканування вразливостей DAST (Dynamic Application Security Testing) – це метод тестування безпеки веб-додатків, що полягає в автоматичному виконанні запитів до програмного забезпечення й аналізі його відповіді на виявлення потенційних вразливостей методом «чорної скриньки» [6]. API-сканування – це процес перевірки безпеки API-інтерфейсів за методом «білої скриньки». Це важлива складова в процесі розробки програмного забезпечення, оскільки API-інтерфейси є ключовими елементами багатьох застосунків та веб-сайтів, тому їх несправність чи компрометація можуть мати серйозні наслідки для безпеки інформації та користувачів [7].

З метою створення системи для виявлення вразливостей програмного забезпечення на всіх стадіях розробки було створено застосунок командного рядка. Зокрема, передбачено підтримку двох типів сканування, які б аналізували весь проект, виявляли придатні для перевірки файли вихідного коду та стандартних бібліотек, для щонайменше 10 мов програмування, як-от Java, .NET, Python, Swift, Go, PHP, JavaScript, C, C++, Ruby, з метою виявлення у них вразливостей безпеки. Програмний продукт розроблено з метою автоматизації процесу виявлення недоліків та прогалин безпеки у користувацьких додатках і інтеграції його в CI/CD. Для цього використовуються декілька типів сканування, які взаємодіють між собою, щоб забезпечити більш широкий спектр інформації про вразливості додатка. Зручність використання програмного продукту забезпечується тим, що всі типи сканування можна запустити в одному застосунку. Застосунок командного рядка може працювати на операційних системах Linux, macOS, Windows, а веб-застосунок підтримується всіма сучасними браузерами. Загалом в основі програми лежить алгоритм, схему якого зображено на рис. 1.

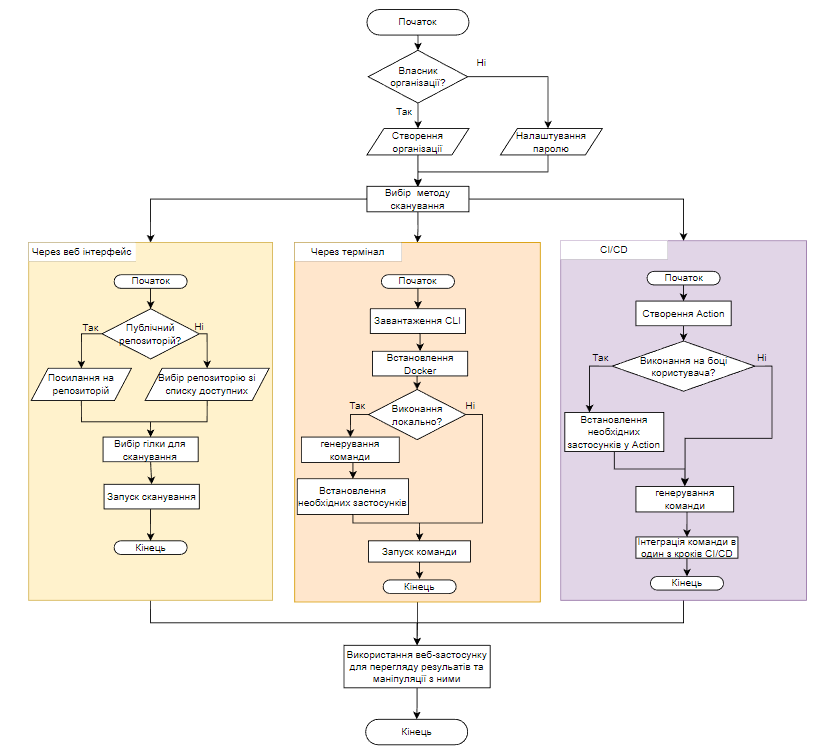


Рис. 1. Загальна схема алгоритму роботи сканеру вразливостей

Після авторизації користувач має змогу інтегрувати через OAuth найпопулярніші систем контролю версій, такі як GitHub, GitLab, Bitbucket і запускати сканування кліками миші з веб інтерфейсу, користувачу надається можливість сканувати публічні репозиторії, або приватні, до яких в нього є доступ, також є можливість сканування різних гілок одного проекту. Сканування різних гілок на предмет вразливостей забезпечує більш комплексний погляд на стан безпеки в різних середовищах, допомагає виявляти вразливості на ранніх етапах розробки.

Однією з ключових характеристик сканерів вразливостей є результат, який відображається користувачеві після завершення сканування, тому у роботі ми перевірили правильність роботи сканеру, його швидкодію в порівнянні з аналогами та протестували основні функції які надаються користувачам, а також визначили рекомендації та обмеження застосування розробки.

Результатом виконання роботи є розроблений сканер для виявлення вразливостей на всіх етапах розробки ПЗ. Завдяки можливості проведення комплексного сканування ПЗ на пристрої користувача або в CI/CD ще на стадії його створення, виконана розробка дає змогу ідентифікувати вразливості системи та рекомендувати можливі способи виправлення чи усунення вразливості на ранніх етапах виготовлення програмного продукту. Створений програмний комплекс може бути застосований для перевірки програмних продуктів, створених найпопулярнішими мовами програмування, в умовах наявності підключення до мережі інтернет і виконання вимог для відповідного типу сканування та мови, та забезпечує 4 різні типи сканування: SCA, SAST, DAST, API, які при спільному використанні відображають реальний стан додатку.

**ЛІТЕРАТУРА**

1. OWASP [Електронний ресурс]. – Режим доступу до ресурсу: <https://owasp.org/>
2. OWASP Top Ten [Електронний ресурс]. – Режим доступу до ресурсу: <https://owasp.org/www-project-top-ten/>
3. The Cost of Finding Bugs Later in the SDLC. [Електронний ресурс]. – Режим доступу до ресурсу: [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.functionize.com/blog/the-cost-of-finding-bugs-later-in-the-sdlc>
4. Software Composition Analysis (SCA): What You Should Know [Електронний ресурс]. – Режим доступу до ресурсу: - https://www.aquasec.com/cloud-native-academy/supply-chain-security/software-composition-analysis-sca/
5. K. Scarfone and P. Mell, "Guide to Static Application Security Testing (SAST)," National Institute of Standards and Technology, 2012.
6. Dynamic Application Security Testing. [Електронний ресурс]. – Режим доступу до ресурсу: - <https://owasp.org/www-community/Testing_Guide_v4.1/contents/11_Testing_Web_Application_Security.html#dynamic-application-security-testing-dast>
7. OWASP API Security Top 10. [Електронний ресурс]. – Режим доступу до ресурсу: - https://owasp.org/Top10/A12\_2021-API\_Security\_Top\_10.html