АНАЛІЗ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ІНФОРМАЦІЇ В ЖИТТЄВОМУ ЦИКЛІ РОЗРОБКИ ПРОГРАМНИХ WEB ПРОДУКТІВ

Качан В.Є., Марчук А.В.

Кафедра інфокомунікаційної інженерії ім. В.В. Поповського,
Харківський національний університет радіоелектроніки,
Україна.

E-mail: vadym.kachan@nure.ua
artem.marchuk@nure.ua

Abstract

In the modern world, web products have taken an important place in people's lives. Thousands of online websites and applications serve customers around the world, raising questions about the security of these products. A necessary aspect of its provision is the use of appropriate solutions in the development and implementation process. This work identifies the need to use a safe development life cycle and the corresponding risks of its absence. A laboratory setup was developed that allows automating most stages of SSDLC and simulates the processes of source code version control, quality control, application assembly and deployment, launch of vulnerability scanners, etc.

Безпека є важливою частиною будь-якого додатку, що містить критичні дані або функції. Використання комплексного підходу, що забезпечить впровадження та реалізацію механізмів безпеки на кожному етапі життєвого циклу розробки (Software Development Life Cycle, SDLC), є необхідною складовою в процесі створення такого додатку. У випадку правильної та раціональної імплементації цих засобів, проблеми в безпеці можуть бути виявлені та виправлені до того як додаток буде відправлено «у виробництво» (deployment in production environment). Безпечний цикл розробки (Secure SDLC, SSDLC) має на меті покласти відповідальність за впровадження відповідних рішень захисту коду від недоліків та вразливостей безпеки безпосередньо на розробників, які мають не тільки створювати функціонал, але й бути впевненими у тому, що дані користувачів та будь-яка конфіденційна інформація захищена відповідним чином. Сьогодні мережа Інтернет наповнена значною кількістю веб-сайтів, веб-додатків та інших продуктів, що надають певні послуги користувачам або надають певні можливості. Серед таких: Інтернет-магазини, додатки банківських та фінансових структур, поштові онлайн-сервіси та сервіси, що надають адміністративні послуги, тощо. Для подібних сервісів відсутність процесу SSDLC може стати критичною та призвести до значних фінансових та репутаційних втрат, тому цей процес є важливим етапом розробки, адже він допомагає ідентифікувати вразливості та запобігти можливим атакам.

Окремою критичною складовою, яка забезпечується використанням безпечного циклу розробки, є відповідність продукту чинному законодавству та стандартам, що відповідають за безпеку. Порушення цих норм може також змусити компанію-розробник понести фінансові втрати або навіть бути притягнутою до відповідальності. Додатково, SSDLC дає не тільки технічне розуміння аспектів впровадження безпеки, а й забезпечує корпоративну дисципліну. У випадку, коли кожний розробник, тестувальник та інженер DevOps розуміє свою роль у забезпеченні безпеки продукту, що розробляється, значною мірою знижуються вищезазначені ризики. Безпечний цикл розробки впроваджується на кожному етапі розробки, таким чином знижуючи відповідні ризики того, що недолік або вразливість, що не була виявлена, залишиться і на подальших етапах. Згідно звіту компанії AutoRabit Software, яка створює DevSecOps рішення для розробників, чим пізніший етап SDLC, тим більше коштів необхідно витратити для усунення вразливості на даному етапі (рис.1) [1]. З аналіза графіка (рис. 1) випливає, що, усунення недоліку на етапі визначення вимог коштує більш ніж в 6 разів дешевше, ніж на етапі підтримки. Досить ефективним також є усунення проблем безпеки на етапі створення продукту. Таким чином, найбільшу увагу слід приділити знаходженню відповідних безпекових рішень для перших двох етапів створення програмного продукту.

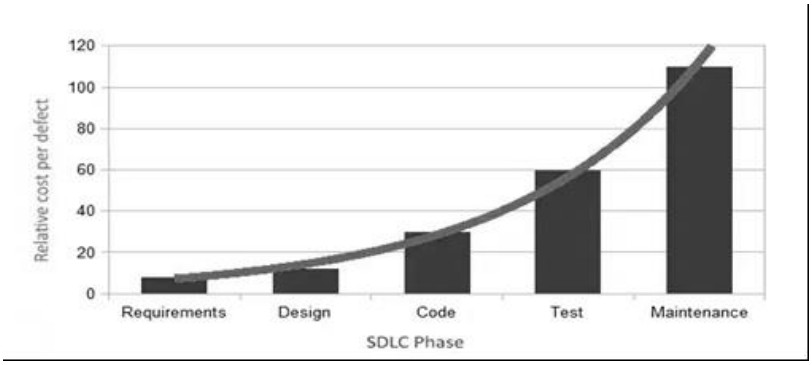

Рис. 1. Витрати на усунення вразливості на різних етапах SDLC

Для вирішення задачі аналізу забезпечення захисту інформації в життєвому циклі розробки програмних продуктів використовується підхід DevSecOps, що складається з ряду безпекових перевірок. На рисунку 2 зображено механізми захисту, що є доцільно використовувати як на етапі розробки (Dev), так і на етапі виконання операцій (Ops) [2].

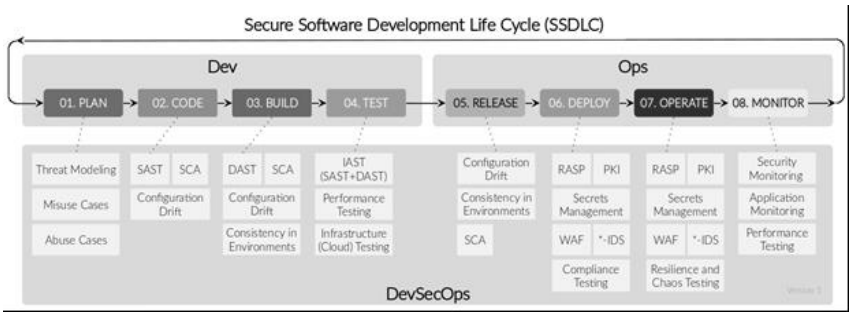

Рис. 2. Механізми забезпечення безпеки на кожному етапі SSDLC

Була розроблена лабораторна установка, що дозволяє автоматизувати більшість етапів SSDLC та моделює процеси контролю версій вихідного коду, перевірки його якості, збірки та розгортання додатку, запуску сканерів вразливостей та інше.

Вихідний код потенційно вразливого додатку зберігається у Git репозиторію системи контролю версій коду. Jenkins виступає як централізований інструмент управління процесами CI/CD, координуючи роботу різних компонентів системи. Він налаштований слідкувати за змінами у Git репозиторії і при кожній новій зміні коду (коміті) запускає спеціально сконфігуровані пайплайни для збірки додатку, сканування якості коду (наприклад SonarQube) та пошуку вразливостей, а також розгортання додатку у різних середовищах тестування та етичного хакінгу.

Під час збірки створюється Docker образ додатку, з якого потім запускаються контейнери для розгортання додатку на різних етапах тестування та впровадження у роботу. Для зберігання різних версій образів використовується Nexus - популярний інструмент управління артефактами та репозиторій програмних продуктів. При необхідності можна робити збірки та тестування різних середовищ, які можуть включати в собі тестові сервери, бази даних, REST-сервіси та інші складові web додатку. Ця лабораторна установка окрім її використання у навчальному процессі дозволяє проводити різноманітні дослідження ефективності різних інструментів виявлення вразливостей, а також різних заходів безпеки, що впроваджуються у SSDLC.

Література

1. Rank J. Estimating ROI with CodeScan [Електронний ресурс] / Josh Rank. – 2018. – Режим доступу до ресурсу: https://www.codescan.io/blog/estimating-roi/.

2. Security along the SDLC for Cloud-Native Apps [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://holisticsecurity.io/2020/02/10/security-along-the-sdlc-for-cloud-native-apps/.