УДК 004.056
DOI: 10.31673/2409-7292.2021.020910

Горюк Н.В., Лавровський І.М.

СТАТИЧНИЙ АНАЛІЗ ВИХІДНОГО КОДУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА БАЗІ РІШЕННЯ FORTIFY STATIC CODE ANALYZER

У статті проаналізовано проблему виявлення вразливостей вихідного коду у контексті розробки програмного забезпечення. Проведено аналіз існуючих технологій виявлення вразливостей вихідного коду. Досліджено методи та засоби захисту виявлення вразливостей вихідного коду на базі рішення Fortify Static Code Analyzer. Визначено призначення, основні функції та архітектуру рішення Fortify Static Code Analyzer. На основі досліджень проведених у роботі розроблено варіант процесу статичного аналізу безпеки вихідного коду у контексті циклу життєдіяльності програмного забезпечення. Розроблено рекомендації щодо застосування технології статичного аналізу безпеки вихідного коду.

Ключові слова: Корпоративна інформаційна система, безпека прикладних програм, безпека вихідного коду, вразливості вихідного коду, тестування безпеки вихідного коду.

Вступ

Розробка програмного забезпечення базується на використанні інформаційних технологій, які, у свою чергу, працюють завдяки великої кількості різноманітного програмного забезпечення. Кожна програма складається з послідовності інструкцій, що виконуються автоматично, одна за одною. Загрози, до яких призводить використання вразливого ПЗ, закладаються ще на етапі його розробки, будь то неправильна фільтрація вхідних даних, або неконтрольована можливість віддаленого виконання коду. Усі вони є результатом невірного використання можливостей того чи іншої мови розробки ПЗ або недоліків апаратного забезпечення. Це робить актуальною проблему виявлення, контролю та виправлення вразливостей ПЗ.

Системи виявлення вразливостей ПЗ на етапі його написання дозволяють розробникам виправляти вразливості ще до того, як вони потраплять до користувачів, що суттєво зменшує затрати на їх виправлення. Вартість виправлення вразливостей, що виправляються після етапу розробки буде значно більшою.

Постановка проблеми

Технологія статичного аналізу вихідного коду надає можливість аналізу кодової бази розроблюваного ПЗ на предмет наявності вразливостей або слабких місць у коді, що робить її актуальним рішенням цієї проблеми. Сучасні сканери вразливостей окрім сканування вихідного коду надають набагато більший набір функцій та допомагають знаходити та централізовано контролювати стан безпеки ПЗ та процес виправлення вразливості протягом усього циклу розробки, а також надають рекомендації щодо усування знайдених вразливостей.

Рішення типу Static Application Security Testing (SAST), сканує написаний розробниками вихідний код програми, його структуру та конфігурацію та робить висновок щодо наявності вразливостей засновуючись не тільки на вже відомих вразливих шаблонах коду, але й на основі аналізу діяльності програми під час його компіляції та роботи. Окрім цього рішення SAST, надають можливості контролю за вразливостями протягом усього життєвого циклу ПЗ та засоби інтеграції з інструментами та технологіями розробки.

Розвиток та впровадження даної технології розробниками ПЗ у процес написання коду робить актуальним розробку рекомендацій щодо використання технології статичного аналізу вихідного коду та її застосування у процесі розробки.

Аналіз літературних джерел

Вразливості програмного забезпечення завжди становили проблему при користуванні програмним та апаратним забезпеченням. В останні роки ця проблема набуває більшого масштабу через прискорення темпів розробки коду, скорочення часу між релізами нових версій програмних продуктів, та ширшого використання стороннього коду у проектах. Для комерційних підприємств ПЗ змінило свою роль з допоміжної – як інструменту, що

використовувався для підвищення ефективності своєї діяльності, до центру інновацій, від якого залежить загальна конкурентоспроможність сучасних IT підприємств. Для державних установ роль ПЗ набула важності у справі реалізації технологічних проектів, будь то створення систем технологізації міст [1], адміністративних баз даних, чи інших, важливих для громади інформаційних систем.

Згідно зі звітом компанії Flexera за 2020 рік [2], у програмному забезпеченні 259 вендорів було знайдено 19 954 вразливості. До 16.6% вразливостей було призначено високий рівень критичності, 0.3% – найвищий. 55.3% вразливостей могли бути експлуатовані з віддалених мереж, 32.1% – з локальних мереж. Ці вразливості були знайдені вже на етапі експлуатації ПЗ, що говорить про необхідність випуску нових виправлених версій продуктів їх розробниками. Разом з потребою постійного випуску функціональних оновлень до розроблюваного ПЗ це створює нові вимоги до рішень з виявлення вразливостей у ПЗ, а саме – можливість швидкої та автоматизованої роботи, що не сповільнювала-б розробників при здійсненні аналізу безпеки вихідного коду, а у перспективі й допомагала їм прискорити виправлення виявлених вразливостей.

Показовими є зрівняння двох систем керування базами даних – PostgreSQL (рис. 1) та MySQL (рис. 1.2). Обидві програми отримують оновлення приблизно раз на три місяці, але статистика загальної кількості вразливостей дуже різниться[3,4].

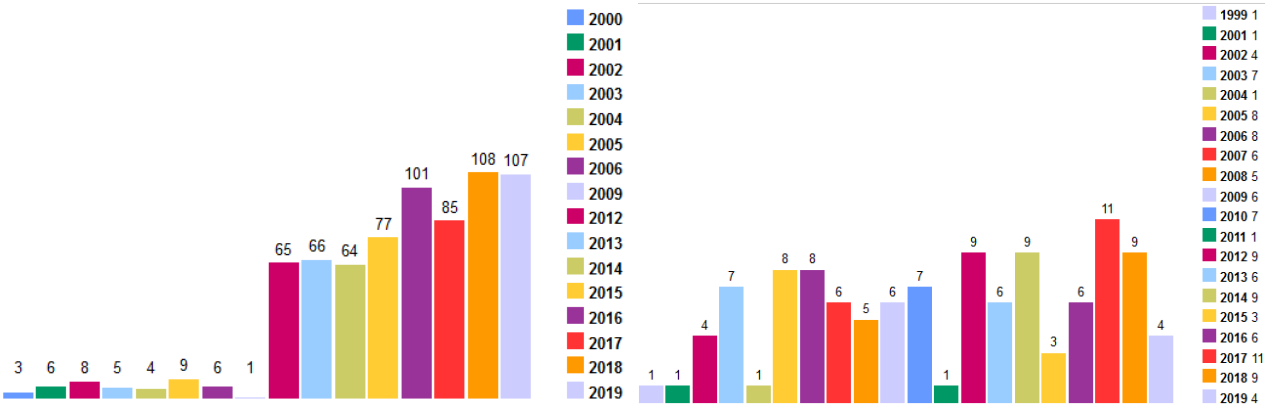

Рис. 1. Статистика вразливостей СКБД MySQL та PostgreSQL по рокам [3,4]

Тобто, у MySQL не тільки з'являється у середньому на рік на 39 вразливостей більше, ніж у postgresql, а й виправляється у середньому, у 8 разів менше вразливостей, ніж у postgresql, враховуючи те, що postgresql містить у 3.5 рази більше строк коду [5,6].

Основні уразливості вихідного коду

Причиною майже кожного інциденту інформаційної безпеки є уразливості інформаційної системи, які використовуються зловмисником задля втручання у нормальну роботу системи. Аналіз повідомлень про вразливості, які проводять такі організації, як CERT або SANS показує [7,8], що багато вразливостей можна звести до невеликої кількісті причин, а саме – одні й ті-ж самі помилки, які роблять розробники програмного забезпечення.

Такі помилки, що призводять до появи вразливостей у безпеці ПЗ можуть бути зведені до наступних п'яти категорій:

Помилки у бізнес-логіці програмного забезпечення.

Помилки вихідного коду.

Відсутність необхідних механізмів безпеки ПЗ.

Вразливість сторонніх компонентів.

Наявність зловмисного коду.

У процесі розвитку технологій та підходів до забезпечення безпеки ПЗ виникла потреба у класифікації вразливостей, завдяки якої можна охарактеризувати виявлені у коді слабкі місця, таким чином дозволяючи розробникам заздалегідь визначити своє ставлення до

вразливостей та скорегувати зусилля на більш значущих. Одну з таких класифікацій погроз надає OWASP [9]:

Ін'єкція коду.
Порушення автентифікації.
Розголошення чутливих даних.
XML External Entity (XXE).
Порушення механізмів контролю доступу.
Неправильна конфігурація механізмів безпеки.
Міжсайтовий скриптинг.
Небезпечна десеріалізація.
Використання вразливих сторонніх компонентів.
Недостатнє журналювання та моніторинг.

Бути впевненим, що розроблений продукт не має недоліків неможливо, однак бізнес завжди ставить задачу знайти баланс між витратами на виявлення та виправлення проблем та прийняттям ризиків. Така ситуація створює необхідність у рішеннях з виявлення та контролю за вразливостями, яке відповідало-б вимогам сфери розробки ПЗ.

Компоненти рішення Fortify Static Code Analyzer та їх призначення

Рішення Fortify Static Code Analyzer версії 20.1.0 складається з декількох компонентів – самого статичного аналізатору коду, та набору плагінів, що відповідають за інтеграцію із середовищем розробки та за ряд додаткових функцій для розробників, та аудиторів безпеки.

Сканер вразливостей вихідного коду являється консольною програмою, що дозволяє легко інтегрувати його у різноманітні середовища розробки на різних операційних системах, у тому числі й у хмарні середовища розробки. Але окрім нього, рішення складається з графічних інтерфейсів та плагінів.

У повний інсталяційний пакет Micro Focus Static Code Analyzer входять наступні компоненти:

Micro Focus Fortify Audit Workbench – графічний інтерфейс користувача для Fortify Static Code Analyzer, який допомагає організовувати, досліджувати та назначати пріоритет виявленим вразливостям, щоб розробники могли швидко виправити недоліки безпеки;

Micro Focus Fortify Custom Rules Editor – інструмент для створення і редагування правил користувача;

Micro Focus Fortify Scan Wizard – інструмент, призначений для швидкої генерації скриптів, які у майбутньому можуть бути використані для автоматизованого сканування вихідного коду;

Утиліти командної строки – набір утиліт командної строки, що встановлюються разом зі сканером вразливостей та мають утилітарне призначення;

Плагін Micro Focus Fortify Plugin for Eclipse – плагін, що додає можливість сканувати та аналізувати всю кодову базу проекту та застосовувати правила безпеки програмного забезпечення, які визначають вразливості вашого коду Java із IDE Eclipse. Також відображає результати та опис до кожної підозрілої ділянки коду та пропозиції щодо їх усунення;

Плагін Micro Focus Fortify Analysis Plugin for IntelliJ та Android Studio –плагін, що додає можливість запускати сканування всієї кодової бази проекту та застосовувати правила безпеки програмного забезпечення, які визначають уразливості вашого коду з інтерактивного середовища розробки IntelliJ та Android Studio;

Micro Focus Fortify Extension for Visual Studio – плагін, що надає можливість сканувати та знаходити дефекти безпеки у рішеннях та проектах, та відображає результати сканування у Visual Studio. Результати включають перелік виявлених проблем, їх описи, та пропозиції щодо їх усунення;

Окрім компонентів, що входять у інсталяційний пакет рішення, існують також розширення до інших технологій, які дозволяють інтегрувати сканер у процес розробки коду.

Micro Focus Fortify Remediation Plugin for Eclipse – плагін встановлюється на IDE Eclipse та інтегрується з Fortify Software Security Center. Його призначено для

централізованого виправлення виявлених вразливостей за допомогою інтеграції з центром безпеки ПЗ;

Micro Focus Fortify Remediation Plugin for JetBrains IDEs – плагін встановлюється на IDE JetBrains та інтегрується з Fortify Software Security Center. Його призначено для централізованого виправлення виявлених вразливостей за допомогою інтеграції з центром безпеки ПЗ;

Micro Focus Fortify Jenkins Plugin – забезпечує можливість здійснення аналізу проекту за допомогою Fortify Static Code Analyzer, а також завантаження результатів аналізу до Fortify Software Security Center, та перегляд результатів у системі Jenkins;

Micro Focus Fortify Bamboo Plugin – забезпечує можливість здійснення аналізу проекту за допомогою Fortify Static Code Analyzer, а також завантаження результатів аналізу до Fortify Software Security Center у системі Bamboo.

Основне призначення аналізатору Fortify SCA – надати розробникам можливість статичного сканування безпеки додатків, для аналізу вихідного коду на предмет наявності вразливих місць. Він переглядає код на предмет наявності у ньому підозрілих місць і допомагає розробникам визначати пріоритет вразливостей, та виправляти ці проблеми з меншими зусиллями та за менший час.

Fortify SCA виконано як консольний додаток задля забезпечення більшого рівня інтеграції та автоматизації процесу сканування вихідного коду. Сам Fortify SCA складається з наступних консольних додатків та утиліт:

sourceanalyzer.exe – консольна утиліта, що використовується для ініціалізації статичного сканування вихідного коду. Налаштування сканування здійснюється за допомогою консольних команд, що надходять при виклику цього файлу. У результаті його роботи створюється спеціальний .fpr-файл, що містить результати аналізу, якщо його було здійснено;

auditworkbench.cmd – скриптовий файл, що запускає графічний інтерфейс Audit Workbench;

BIRTReportGenerator.cmd – скриптовий файл, що використовується для створення файлу звітності Business Intelligence and Reporting Tools (BIRT);

CustomRulesEditor.cmd – скриптовий файл, що запускає графічний інтерфейс Custom Rules Editor, що використовується задля редагування довільних правил виявлення вразливостей;

fortifyclient.bat – консольна утиліта, що використовується задля забезпечення автоматизації деяких дій з файлами, та інтеграції з сервером безпеки ПЗ (Fortify Software Security Server);

fortifyupdate.cmd – консольна утиліта, що використовується задля здійснення оновлення рішення та правил виявлення вразливостей до останніх версій, та автоматизації цього оновлення;

FPRUtility.bat – консольна утиліта, що використовується сканером та користувачами задля здійснення дій над файлами з результатами статичного аналізу, як то їх конвертація та з'єднування. Також може бути використана для здійснення налаштування сканеру вразливостей при взаємодії з цими файлами;

iidmigrator.bat – скриптовий файл, що використовується для конвертації файлів результатів аналізу у новіші версії;

packagescanner.bat – консольна утиліта, що використовується сканером для здійснення функцій інтеграції з хмарним середовищем сканування (Fortify ScanCentral), як то сканування пакетів з кодом сканованого проекту;

pwtool.bat – консольна утиліта, що використовується при інтеграції локального сканеру вихідного коду з хмарним середовищем сканування (Fortify ScanCentral), як то генерація ключів доступу для забезпечення безпечної інтеграції;

ReportGenerator.bat – консольна утиліта, що використовується як користувачами так і самим сканером вихідного коду задля створення звітів по результатам здійсненного сканування, наприклад у форматі PDF застарілого зразку;

scancentral.bat – консольна утиліта що використовується у системі хмарного середовища сканування задля здійснення його запуску, налаштування та експлуатації. Це середовище використовується для розподілення обчислювальних ресурсів інформаційної мережі підприємства при статичному аналізі вихідного коду проекту;

ScanWizard.cmd – скриптовий файл, що запускає графічний інтерфейс ScanWizard – утиліти, що допомагає розробникам створити скриптові файли, які містять команди запуску сканування на основі вказаного проекту. Використовується для автоматизації, спрощення та прискорення процесу статичного аналізу вихідного коду;

scapostinstall.cmd – консольна утиліта, що за допомогою файлів конфігурації минулих версій Fortify SCA дозволяє застосувати ті-ж налаштування до оновленої версії рішення;

SCAState.cmd – скриптовий файл, що надає можливість переглянути стан поточного сканування.

Рішення Fortify SCA тісно взаємодіє з інструментами розробки програмного забезпечення, та за допомогою вбудованих засобів та утиліт здатне автоматизовано здійснювати статичне сканування вихідного коду. Окрім того, набір плагінів дозволяє рішенню здійснювати безшовну інтеграцію з популярними інтегрованими середовищами розробки. Інтеграція з центром безпеки ПЗ Fortify SSC надає можливість використовувати сканер вразливостей нарядом з корпоративними системами авторизації та службами каталогів, що поширює її налаштування безпеки. Більш повне зображення екосистеми рішень Fortify можна побачити на діаграмі нижче (рис. 2).

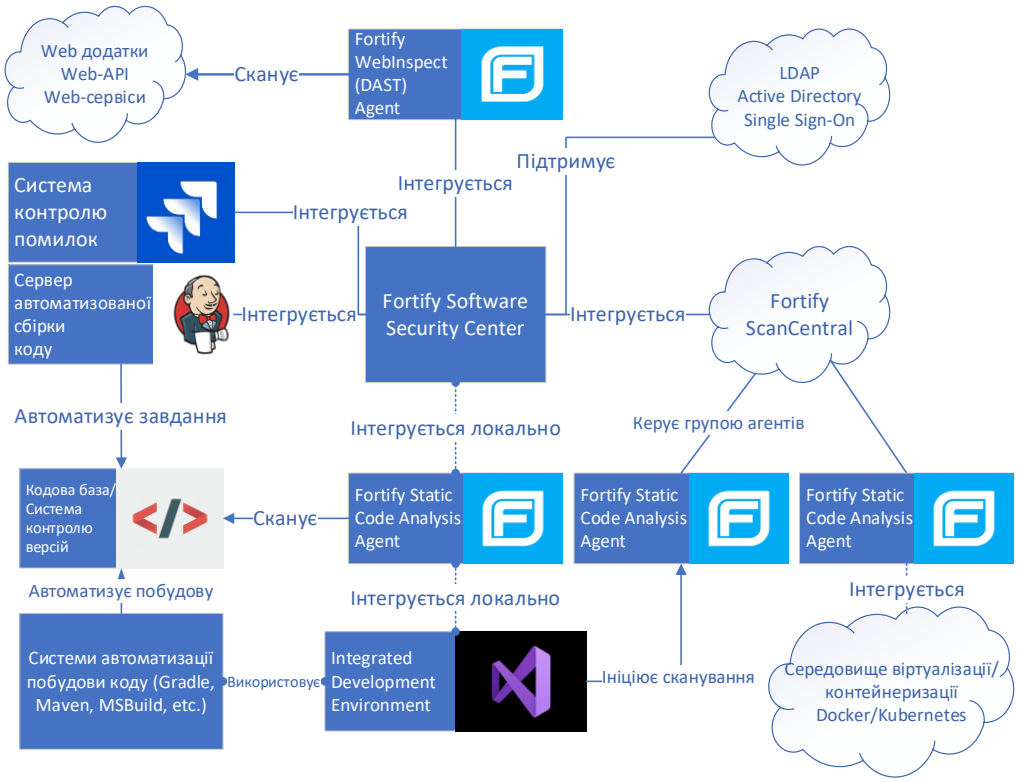

Рис. 2. Система рішень Fortify та інструментів розробки коду

Алгоритм статичного аналізу вихідного коду ПЗ

Контроль за вразливостями у коді здійснюється у контексті процесу розробки коду, що є унікальним для кожного підприємства. Все це ускладняється динамічністю ринку технологій розробки, що згодом впроваджує нові засоби та інструменти до процесу

розробки. Так, рішення, що використовується задля забезпечення аналізу безпеки коду повинно бути здатним не тільки до інтеграції з актуальними технологіями розробки коду, а й активно отримувати оновлення як бази вразливостей, так і власних засобів інтеграції.

Fortify SCA здатен до аналізу як вихідного так і бінарного коду. Такий набор функціональних можливостей дозволяє поширити покриття етапів розробки ПЗ рішенням. Більше детальніше покриття Fortify процесу розробки зображено нижче (рис. 3).

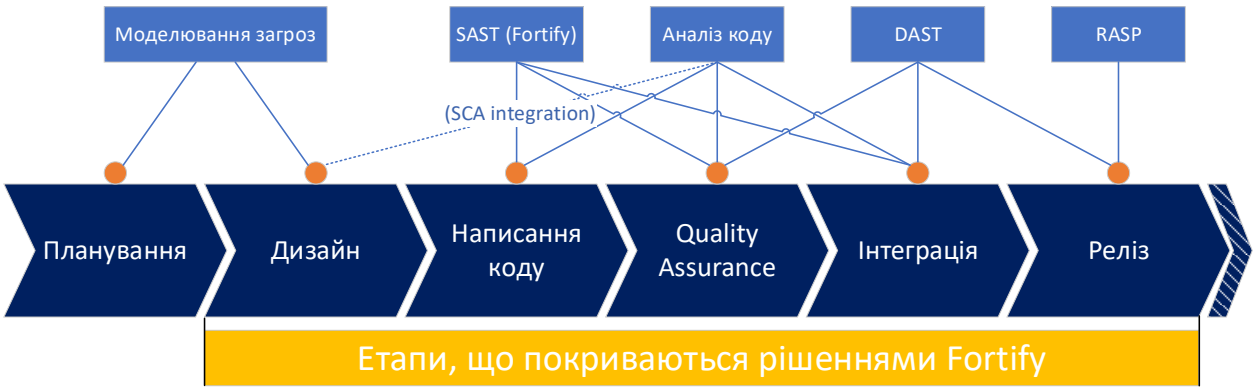

Рис. 3. Покриття етапів розробки рішеннями Fortify

Тому як процес розробки, використані технології і інструменти у кожному підприємстві можуть різнитися – процес аналізу безпеки також буде різним. Від того, у якому вигляді проходить процес розробки ПЗ, залежить те, яким чином буде виконуватися процес контролю за вразливостями, а значить й процес контролю вразливостей вихідного коду.

Однак, Micro Focus має стандартизований підхід до здійснення процесу аналізу коду рішенням Fortify SCA, який у разі потреби може бути змінено – скорочено або доповнено за умови, що етапи аналізу та аудиту будуть збережені (рис. 4).

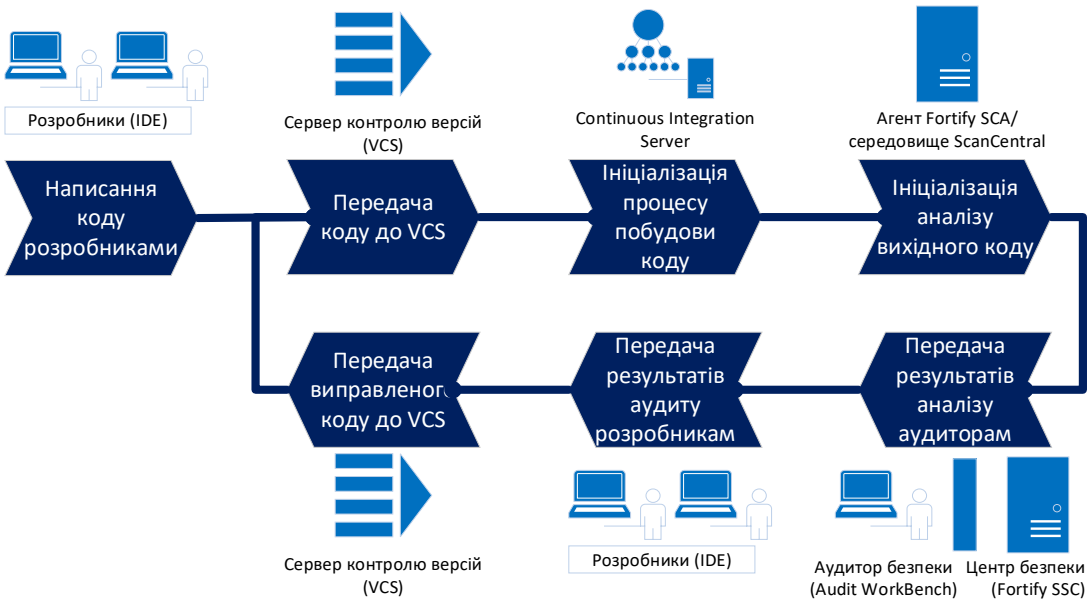

Рис. 4. Алгоритм здійснення статичного аналізу вихідного коду рішенням Fortify SCA

Після стадії початкового планування проект переходить до стадії дизайну, частиною якого є створення функціонального ескізу ПЗ. Функціональний ескіз продукту хоч і не вміщає основної долі кодової бази та його сканування не може вважатися за повноцінний аналіз безпеки, але необхідні залежності, що у майбутньому будуть використані при розробці коду у ньому вже присутні. Такими залежностями можуть бути сторонні компоненти як-то

відкритий код, бібліотеки, комерційні компоненти тощо. Сторонні компоненти використовуються неодноразово та інформація щодо вразливих версій компонентів нерідко публікуються у переліках вразливих компонентів (CVE). Окрім цих баз існують також комерційні рішення, що надаюсь власні можливості сканування списку сторонніх компонентів задля виявлення та виключення з розробки їх вразливих версій. Це також називається композиційним аналізом. Прикладом рішення, що надає такий набір функцій є WhiteSource [10].

Таким чином можна виявляти, аналізувати, пріоритезувати та виправляти вразливості у коді, не порушуючи процес розробки. Використання засобів автоматизації дозволяє уникнути потреби у участі розробників та спеціалістів безпеки у процесі технічного аналізу коду, що дозволяє не тільки не затрачати їх робочий час на завдання технічного характеру, а й прискорити процес аналізу коду [11].

Висновки

У роботі запропоновано варіант процесу здійснення аналізу безпеки вихідного коду на базі рішення Fortify Static Code Analyzer. У ході виконання задачі аналізу безпеки вихідного коду проводиться ряд операцій з інтеграції рішення з технологіями та інструментами, що використовуються при розробці коду та продемонстровано автоматизований процес технічного аналізу безпеки вихідного коду та його проміжні результати.

Реалізація запропонованих рекомендацій з впровадження, інтеграції та використання технології статичного аналізу безпеки ПЗ на базі рішення Fortify Static Code Analyzer має доповнити процес розробки ПЗ ефективними методами автоматизованого виявлення, контролю та виправлення вразливостей.

Перелік посилань

1. Kyiv Smart City [Електронний ресурс] / Автори Вікіпедії // Версія 27695562. – 2020. – Режим доступу: World Wide Web. - URL: https://uk.wikipedia.org/w/index.php?title=Kyiv_Smart_City&oldid=27695562.
2. Vulnerability Review 2020 Global Trends [Електронний ресурс]. – 2020. – Режим доступу: World Wide Web. - URL: https://resources.flexera.com/web/pdf/Report-SVM-Vulnerability-Review-2020.pdf.
3. CVE Details - Postgresql Vulnerability Statistics [Електронний ресурс] – Режим доступу: World Wide Web. - URL: https://www.cvedetails.com/product/575/Postgresql-Postgresql.html?vendor_id=336.
4. CVE Details - Oracle Mysql Vulnerability Statistics [Електронний ресурс] – Режим доступу: World Wide Web. - URL: https://www.cvedetails.com/product/21801/Oracle-Mysql.html?vendor_id=93.
5. Google Summer of Code project ideas 2020, Improve PostgreSQL Regression Test Coverage [Електронний ресурс]. – 2020. – Режим доступу: World Wide Web. - URL: https://wiki.postgresql.org/wiki/GSoC_2020.
6. The MySQL Open Source Project on Open Hub: Language Page [Електронний ресурс] – Режим доступу: World Wide Web. - URL: https://www.openhub.net/p/mysql/analyses/latest/languages_summary.
7. SANS Institute | Top 25 Software Errors [Електронний ресурс]. – 2020. – Режим доступу: World Wide Web. - URL: https://www.sans.org/top25-software-errors/.
8. CERT Secure Coding - Top 10 Secure Coding Practices [Електронний ресурс]. – 2018. – Режим доступу: World Wide Web. - URL: https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices.
9. OWASP Top 10 Application Security Risks - 2017 [Електронний ресурс]. – 2017. – Режим доступу: World Wide Web. - URL: https://owasp.org/www-project-top-ten/2017/Top_10.html.
10. The WhiteSource Solution [Електронний ресурс] // WhiteSource. – 2017. – Режим доступу: World Wide Web. - URL: https://resources.whitesourcesoftware.com/product-datasheets/the-whitesource-solution.
11. Горюк Н. В. Засоби інтеграції технології статичного аналізу безпеки вихідного коду у середовище розробки програмного забезпечення / Горюк Н. В. // «Сучасний захист інформації» – 2020 – №3. – С.54-58.

Надійшла: 08.04.2021 Рецензент: д.т.н., професор Савченко В.А.