Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Дослідження методів забезпечення безпеки ланцюжків поставок програмного забезпечення
(тема)

Виконала:
студентка 2 курсу, групи АМСЗІзм-22-1
Новікова Д.О.
(прізвище, ініціали)

Спеціальність: 125 Кібербезпека
(код і повна назва спеціальності)
Тип програми: освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма: Адміністративний менеджмент у сфері захисту інформації
(повна назва освітньої програми)

Керівник: професор кафедри ІКІ ім. В.В. Поповського
Радівілова Т.А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Лемешко О.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Адміністративний менеджмент у сфері захисту інформації .
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри_____
(підпис)
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Новіковій Діані Олексіївні .
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів забезпечення безпеки ланцюжків поставок програмного забезпечення
.
затверджена наказом по університету від «06» листопада 2023 р. №246 Стз.
2. Термін подання студентом роботи до екзаменаційної комісії 30.01.2024 р
.
3. Вихідні дані до роботи: етапи розробки програмного забезпечення, методи забезпечення безпеки на кожному етапі розробки програмного забезпечення, технологія перевірки ланцюжка постачання програмного забезпечення, атаки на ланцюжки постачання програмного забезпечення .
4. Перелік питань, що потрібно опрацювати в роботі:
1) Аналіз технологій перевірки ланцюжка постачання програмного забезпечення
.
2) Аналіз атак на ланцюжки постачання програмного

забезпечення .

3) Існуючі методи забезпечення безпеки на кожному етапі розробки програмного забезпечення .

4) Дослідження методів забезпечення безпеки на кожному етапі розробки програмного забезпечення .

5. Перелік графічного матеріалу із зазначенням креслень, плакатів, комп'ютерних ілюстрацій: демонстраційний матеріал у вигляді ppt-презентації .

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Основна частина	професор Радівілова Тамара Анатоліївна		05.12.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.09.2023	Виконано
2	Збір матеріалів для дослідження	16.09.2023	Виконано
3	Розробка 1 розділу	21.09.2023	Виконано
4	Розробка 2 розділу	12.10.2023	Виконано
5	Розробка 3 розділу	25.11.2023	Виконано
6	Розробка 4 розділу	05.12.2023	Виконано
7	Оформлення кваліфікаційної роботи	15.12.2023	Виконано

Дата видачі завдання 01 вересня 2023 року .

Студентка  Новікова Д.О.
(підпис) (прізвище, ініціали)

Керівник роботи  професор Радівілова Т.А.
(підпис) (посада, прізвище, ініціали)

Робота не містить відомостей заборонених до відкритого

опублікування

Студент  **Діана НОВІКОВА**

Керівник  **Тамара РАДІВІЛОВА**

РЕФЕРАТ

Пояснювальна записка: 85 с., 7 рис., 8 табл., 25 джерел.

АТАКИ, ВРАЗЛИВОСТІ, ЕТАПИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЗАГРОЗИ, ЗАХИЩЕНІСТЬ ЛАНЦЮЖКІВ ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ, МОДЕЛЬ ЗАХИСТУ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РИЗИКИ, ТЕХНОЛОГІЇ ПЕРЕВІРКИ.

Об’єкт дослідження – процес забезпечення безпеки ланцюжків постачання програмного забезпечення.

Предмет дослідження – методи та засоби забезпечення безпеки ланцюжків постачання програмного забезпечення.

Мета роботи – аналіз шляхів збільшення ефективності методів забезпечення безпеки ланцюжків постачання програмного забезпечення на кожному етапі розробки програмного забезпечення.

Методи досліджень – моделювання, спостереження, аналіз та порівняння.

Сьогодні суть розробки програмного забезпечення полягає в тому, що постачальники та розробники програмного забезпечення часто створюють продукти, збираючи компоненти комерційного програмного забезпечення з відкритим кодом, проводять його тестування та якнайшвидше розгортають його у робочому середовищі. У той час як швидкість розгортання є ключовим пріоритетом, безпека ланцюжка постачання програмного забезпечення – не менш важлива для будь-якої організації.

Зловмисники використовують сучасні гіперз’єднані ланцюжки постачання програмного забезпечення, оскільки один «пролом» дозволяє їм отримати доступ до великої кількості систем, що знаходяться нижче. Це означає, що команди безпеки та організації-розробники у всьому світі повинні ставити безпеку ланцюжка постачання на перше місце [2].

У роботі було проведено дослідження методів забезпечення безпеки ланцюжків постачання програмного забезпечення. Особлива увага приділена забезпеченню безпеки на кожному етапі розробки програмного забезпечення.

Виходячи з того, що кожен крок ланцюжка постачання програмного забезпечення є вразливим до атак, його слід перевіряти на кожному кроці процесу, включаючи процес розробки програмного забезпечення.

Розроблено модель захисту ланцюжків постачання програмного забезпечення.

ABSTRACT

The report contains: 85 p., 7 fig., 8 tables, 25 sources.

ATTACKS, VULNERABILITIES, SOFTWARE DEVELOPMENT STAGES, THREATS, SOFTWARE SUPPLY CHAIN SECURITY, SECURITY METHODS, SECURITY MODEL, SOFTWARE, RISKS, VERIFICATION TECHNOLOGIES.

A research object is the process of ensuring the security of software supply chains.

The subject of the research is methods and facilities of ensuring the security of software supply chains.

An aim of work is an analysis of ways of increasing the effectiveness of software supply chain security methods at each stage of software development.

Methods of researches are modeling, observation, analysis, and comparison.

Today, the nature of software development is that software vendors and developers often build products by assembling components of commercial open-source software, testing it, and deploying it to the production environment as quickly as possible. While speed of deployment is a key priority, the security of the software supply chain is equally important for any organization.

Attackers are exploiting today's hyper-connected software supply chains because a single breach allows them to access a large number of systems downstream. This means that security teams and development organizations around the world must prioritize supply chain security [2].

This paper investigates methods of ensuring the security of software supply chains. Particular attention is paid to ensuring security at each stage of software development.

Since every step of the software supply chain is vulnerable to attack, it should be checked at every step of the process, including the software development process.

A model is developed for software supply chain security.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІБ – інформаційна безпека
ІКТ – інформаційно-комунікаційні технології
ІС – інформаційна система
ОС – операційна система
ОТ – операційні технології
ПЗ – програмне забезпечення
СУБД – система управління базами даних
ТЗІ – технічний захист інформації
API – Application Programming Interface
ASVS – Application Security Verification Standard
CI/CD – Continuous Integration / Continuous Delivery or Continuous
DAST – Dynamic Application Security Testing
DevSecOps – Development, Security, Operations
GDPR – General Data Protection Regulation
GSD – Global Software Development
IDE – Integrated Development Environment
IT – Information Technology
MFA – Multi-factor Authentication
MSP – Managed Service Providers
NPM – Network Performance Monitor
NPM – Network Performance Monitoring
OWASP – Open Web Application Security Project
SAST – Static Application Security Testing
SBOM – Software Bill of Materials
SCA – Software Composition Analysis
SDLC – Software Development Life Cycle
SLSA – Supply-chain Levels for Software Artifacts
SQL – Structured Query Language
TRM – Third Party Risk Management
XSS – Cross-Site Scripting

ВСТУП

Кібербезпека ланцюжків постачання вважається невід'ємною частиною заходів, що стосуються управління ризиками кібербезпеки.

Основна увага приділяється ланцюжкам постачання інформаційно-комунікаційних технологій (ІКТ) та операційних технологій (ОТ). Передова практика повинна надаватися та може бути реалізована клієнтами (наприклад, організаціями, визначеними як важливі та важливими суб’єктами) або їх відповідними постачальниками та провайдерами.

Передова практика забезпечення кібербезпеки ланцюжків постачання повинна охоплювати п'ять областей, а саме:

– стратегічний корпоративний підхід;
– управління ризиками ланцюжка постачання;
– управління взаємовідносинами із постачальниками;
– обробку вразливостей;
– якість продукції та практики постачальників та провайдерів послуг.

Тобто, організаціям необхідно створити загальнокорпоративну систему управління ланцюжками постачання, засновану на управлінні ризиками третіх сторін TRM (Third Party Risk Management) і охоплюючу оцінку ризиків, управління взаємовідносинами з постачальниками, управління вразливостями та якістю продукції.

Така передова практика повинна охоплювати всі організації, що відіграють роль у ланцюжку постачання продуктів та послуг ІКТ/ОТ, від виробництва до споживання. Необхідно враховувати, що не всі сектори демонструють однакові можливості щодо управління ланцюжками постачання ІКТ/ОТ.

Опитування Всесвітнього економічного форуму (WEF) і Anchore повідомляють, що від 39 % до 62 % організацій постраждали від кіберінциденту третьої сторони [10]. Крім того, згідно з Mandiant, порушення ланцюжка постачання були другим за поширеністю початковим вектором зараження, виявленим у 2021 році. На них також припадає 17 % вторгнень у 2021 році порівняно з менш ніж 1 % у 2020 році [12].

У 2021 році ENISA’s Threat landscape для атак ланцюжка постачання показує, що в 66 % аналізованих атак на ланцюжки постачання постачальники не знали або не були прозорими щодо того, як вони були скомпрометовані. Навпаки,

менше 9 % клієнтів, скомпрометованих через атаки на ланцюжок постачання, не знали, як відбулися атаки. Це підкреслює розрив у звітності про інциденти кібербезпеки між постачальниками та компаніями, що працюють із кінцевими користувачами. Близько 62 % атак на клієнтів було проведено шляхом зловживання їх довірою до свого постачальника. У 62 % випадків використаним методом атаки було зловмисне програмне забезпечення. Розглядаючи цільові активи, у 66 % інцидентів зловмисники зосереджувалися на коді постачальників, щоб ще більше скомпрометувати цільових клієнтів.

В звіті ENISA's Threat landscape (2022) також спостерігається підвищений інтерес груп загроз до атак на ланцюжки постачання і атак на постачальників керованих послуг MSP (Managed Service Providers) [4].

Таким чином, задача щодо вибору (розробки) методики забезпечення захищеності ланцюжків постачання програмного забезпечення з метою запобігання використання зловмисниками існуючих загроз та вразливостей для реалізації атак на організацію; задача вибору методів мінімізації ризиків ланцюжків постачання та оцінки рівня захищеності інформаційної системи організації від атак даного типу є вкрай важливими не тільки науковими, але й практичними задачами.

Метою даної роботи є розробка методики забезпечення захищеності ланцюжків постачання, включаючи безпеку на кожному етапі розробки програмного забезпечення на основі моделі захисту ланцюжків постачання програмного забезпечення.

Для вирішення поставленої задачі в першому розділі було проаналізовано технології перевірки ланцюжка постачання програмного забезпечення. Було розглянуто типи атак на ланцюжки постачання програмного забезпечення; методи пом'якшення загроз ланцюжка постачання; методи безпеки, які слід враховувати командам безпеки, зокрема, методи безпечного кодування, використання файлів блокувань та інші ініціативи, орієнтовані на безпеку; розглянуто ланцюжок постачання в NIS2 Directive та цикл PDCA для кібербезпеки ланцюжка постачання.

В другому розділі було проведено аналіз атак на ланцюжки постачання програмного забезпечення. Розглянуто, статистику атак на ланцюжок постачання програмного забезпечення, останні відомі атаки даного типу та методи зменшення ризиків безпеки.

В третьому розділі було проведено дослідження існуючих методів забезпечення безпеки на кожному етапі розробки програмного забезпечення, зокрема, розглянуто життєвий цикл безпечної розробки програмного забезпечення (SSDLC).

В четвертому розділі було розроблено модель захисту ланцюжків постачання програмного забезпечення, зокрема методику забезпечення безпеки на кожному етапі розробки програмного забезпечення, принципи управління ризиками безпеки ланцюжка постачання за фазою SDLC тощо. Створено методику оцінки захищеності інформаційної системи організації від атак на ланцюжки постачання; запропоновано методику, що дозволить коректно оцінити значення ризику інформаційної безпеки та скалькулювати грошові втрати у разі виникнення інцидентів безпеки. Було проведено оцінку ефективності розробленої моделі та методики.

Окремі результати роботи було опубліковано в статті та тезах доповідей на трьох Міжнародних наукових конференціях [1 – 3].

1 АНАЛІЗ ТЕХНОЛОГІЙ ПЕРЕВІРКИ ЛАНЦЮЖКА ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Компоненти, бібліотеки, інструменти та процеси, які використовуються для розробки, створення та публікації артефактів програмного забезпечення є складниками ланцюжка постачання програмного забезпечення [2]. Як відомо, постачальники програмного забезпечення часто створюють продукти, збираючи компоненти комерційного програмного забезпечення з відкритим кодом.

Розглянемо типову структурну схему ланцюжка постачання програмного забезпечення, яка представлена на рисунку 1.1.

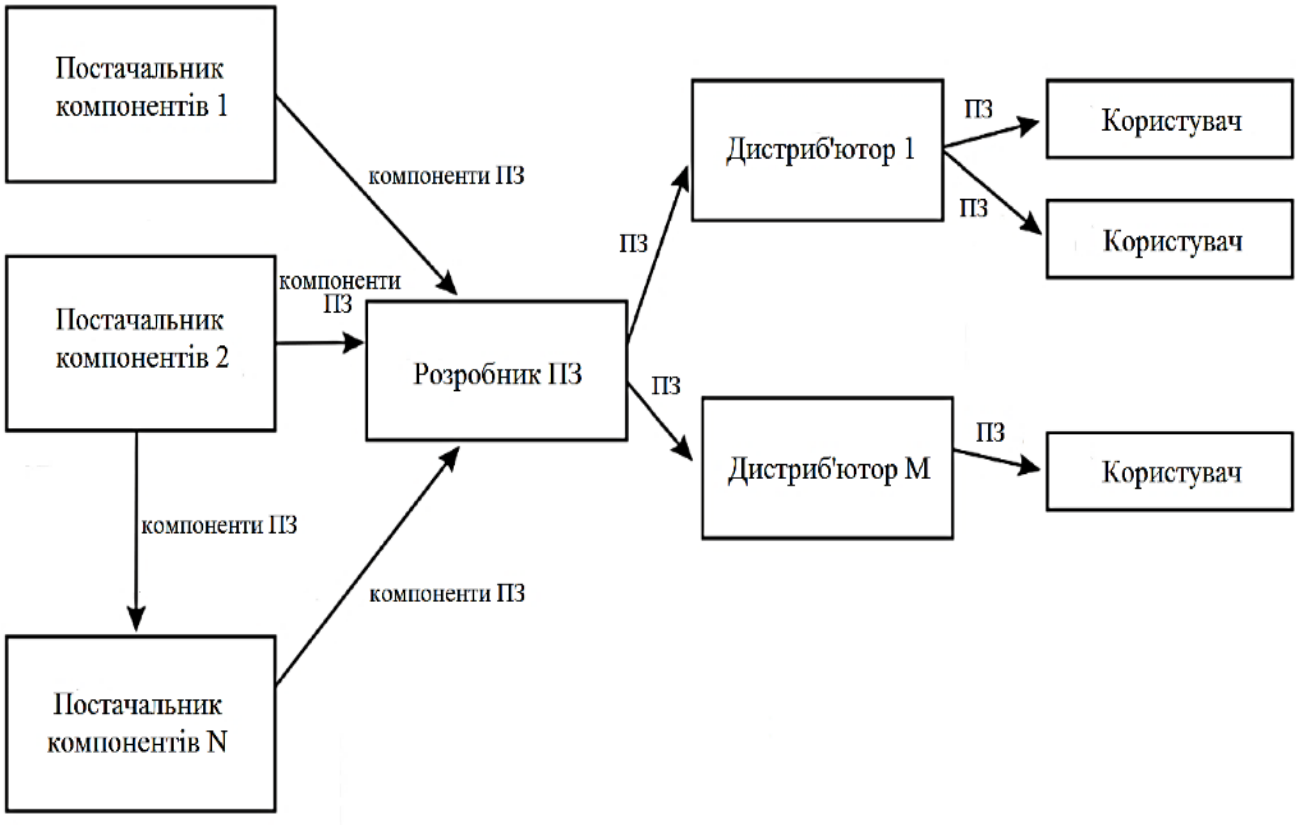

Рисунок 1.1 – Типова структура ланцюжка поставок програмного забезпечення [5]

На рисунку 1.1 можна побачити, що ланцюжок постачання програмного забезпечення (ПЗ) характерно складається з постачальників компонентів ПЗ, розробників ПЗ, дистриб’юторів та користувачів, яких об’єднують довірені відносини. А як відомо, атака на ланцюжок постачання програмного забезпечення

спрямована на довіреного стороннього постачальника, який пропонує ПЗ, що є життєвоважливими для ланцюжка постачання.

Безпека ланцюжка постачання програмного забезпечення повинна поєднувати в собі найкращі практики управління ризиками та кібербезпеки, допомагаючи захистити ланцюжок постачання програмного забезпечення від потенційних вразливостей.

Ланцюжок постачання програмного забезпечення складається з усього і вся, що стосується коду в життєвому циклі розробки програмного забезпечення (SDLC), від розробки програм до конвеєра CI/CD та розгортання. Він включає мережі інформації про програмне забезпечення, наприклад, про компоненти (такі як інфраструктура, обладнання, операційні системи, хмарні сервіси та інше), людей, які їх написали, і джерела, з яких вони надходять, наприклад, реєстри, GitHub репозиторії, бази коду чи інші проекти із відкритим вихідним кодом. Крім того, ланцюжок постачання програмного забезпечення включає будь-які уразливості, які можуть негативно вплинути на безпеку програмного забезпечення – і тут на допомогу приходять засоби та методи забезпечення безпеки ланцюжка постачання програмного забезпечення [2].

Безпека ланцюжка постачання програмного забезпечення повинна мати вирішальне значення для будь-якої організації.

Як відомо, більшість програмного забезпечення сьогодні не пишеться з нуля – зазвичай воно є комбінацією програмних артефактів, що містять програмне забезпечення з відкритим вихідним кодом. Однак ці програмні артефакти схильні до вразливостей, і розробники мають менше контролю над вихідним кодом третіх осіб або будь-якими змінами, що вносяться до програмного артефакту з часом. Важливо відзначити, що неоновлене програмне забезпечення більш схильне до проблем безпеки. Оскільки програмне забезпечення має важливе значення для щоденних бізнес-операцій, безпека ланцюжка постачання є найважливішим обов'язком кожної організації та групи безпеки [3].

Компанія-розробник програмного забезпечення SolarWinds зазнала злому в 2020 році, коли зловмисники запустили шкідливий код через її програмне забезпечення для моніторингу та управління IT Orion, що представляє собою платформу, яку використовують великі корпорації та урядові установи. Атакуючи ланцюжок постачання, зловмисники вплинули не лише у SolarWinds, а й на її клієнтів [7].

Для безперервної розробки програмного продукту з урахуванням вимог безпеки існує методика DevSecOps. Вона представляє собою практику інтеграції тестування безпеки на кожному етапі процесу розробки програмного забезпечення.

DevSecOps означає розробку, безпеку та операції. Дана методика є підходом до культури, автоматизації та дизайну платформи, який інтегрує безпеку як спільну відповідальність протягом усього життєвого циклу інформаційних технологій.

DevSecOps означає думати про безпеку програм та інфраструктури з самого початку. Це також вказує на автоматизацію деяких захисних «воріт», щоб уникнути сповільнення робочих процесів DevOps. Вибір правильних інструментів для постійної інтеграції безпеки, наприклад узгодження інтегрованого середовища розробки IDE (Integrated Development Environment) із функціями безпеки, може допомогти досягти цих цілей.

У той час як ланцюжок постачання програмного забезпечення складається з усього і кожного, хто має відношення до конкретного коду, безпека програми захищає сам код від атак і вразливостей. Подібно до безпеки ланцюжка постачання програмного забезпечення, безпека додатків також повинна застосовуватися на кожному кроці розробки.

Безпека програми починається з життєвого циклу розробки програмного забезпечення та поширюється на весь життєвий цикл програми з метою запобігання несанкціонованому доступу до конкретної системи та захисту конфіденційних даних. Зміцнення цілісності ланцюжка постачання може, у свою чергу, підвищити безпеку програм. Зміцнення конфігурацій, мінімізація поверхонь для атак, обмеження дозволів, підписування програмного забезпечення та розповсюдження збірок у різних частинах системи – усе це є способами захисту від компрометації програм зловмисниками.

Розглянемо методи пом'якшення загроз ланцюжка постачання програмного забезпечення.

Безпека ланцюжка постачання програмного забезпечення важлива для підприємства, клієнтів і будь-якої організації, яка покладається на внески з відкритим кодом. Хоча жодна організація не хоче бути схильна до злому зі сторони зловмисників, вона також не прагне нести відповідальність за іншу організацію, яка зіткнулася з подібною подією. Реалізація засобів захисту для ланцюжка постачання програмного забезпечення є ключовим завданням.

Наведемо деякі найкращі методи безпеки, які слід враховувати командам безпеки.

1) Надавати найменш привілейований доступ до ресурсів у всьому ланцюжку постачання (наприклад, це стосується інструментів розробника, сховищ вихідного коду та інших систем програмного забезпечення), увімкнути багатофакторну автентифікацію та використовувати надійні паролі.

2) Проводити регулярне навчання співробітників з безпеки.

3) Підвищити безпеку всіх підключених пристроїв і конфіденційних даних.

4) Дізнатися про постачальників і про тих, з ким працює конкретна організація, починаючи з постачальників першого рівня; провести оцінку ризиків, щоб оцінити стан кібербезпеки кожного постачальника та дотримуватися державної політики щодо вразливостей.

5) Регулярно сканувати та виправляти вразливі системи.

Розробникам також слід розглянути методи безпечного кодування, використання файлів блокувань та інші ініціативи, орієнтовані на безпеку. Такі ініціативі можуть включати наступні рекомендації.

1) Перевірити контрольні суми.

2) Включити залежності від постачальника в систему керування джерелами.

3) Публікувати та використовувати SBOM – Software Bill of Materials.

4) Використовувати SLSA – Supply-chain Levels for Software Artifacts, що включає:

– можливість цифрового підпису артефактів програмного забезпечення для автентифікації походження;

– використання автоматизації для процесів і політик організації.

5) Сканувати програмне забезпечення за допомогою автоматизованих інструментів тестування безпеки, таких як:

– аналіз складу програмного забезпечення SCA – Software Composition Analysis;

– статичне тестування безпеки додатків SAST – Static Application Security Testing;

– динамічне тестування безпеки додатків DAST – Dynamic Application Security Testing [2].

Все це необхідно враховувати командам безпеки для того, щоб пом'якшити загрози ланцюжка постачання програмного забезпечення.

1.1 Ланцюжок постачання в NIS2 Directive

Директива (ЄС) 2022/2555 (NIS2 Directive) вимагає від держав-членів гарантувати, що основні та важливі суб’єкти вживають належних і пропорційних технічних, операційних та організаційних заходів для управління ризиками, пов’язаними з безпекою мережевих та інформаційних систем, які ці суб’єкти використовують для надання своїх послуг. Кібербезпека ланцюжка постачання вважається невід’ємною частиною заходів з управління ризиками кібербезпеки відповідно до статті 21(2) NIS2 Directive.

Передові практики щодо кібербезпеки ланцюжка постачання, які випливають із європейських і міжнародних стандартів зосереджені насамперед на ланцюжках постачання інформаційно-комунікаційних технологій (ІКТ) або операційних технологій (ОТ). Належні практики надаються та можуть бути впроваджені клієнтами (наприклад, організаціями, визначеними як основні та важливі суб’єкти відповідно до директиви NIS2) або їхніми відповідними постачальниками та провайдерами [9].

Передові практики охоплюють п’ять сфер, а саме: стратегічний корпоративний підхід; управління ризиками ланцюжка постачання; управління взаємовідносинами з постачальниками; усунення вразливостей; якість продукції та практики для постачальників і провайдерів послуг.

Наведемо деякі висновки щодо ланцюжка постачання ІКТ/ОТ.

1) Існує плутанина щодо термінології навколо ланцюжка постачання ІКТ/ОТ.
2) Організації повинні створити загальнокорпоративну систему управління ланцюжком постачання, яка базується на управлінні ризиками третьої сторони (TRM) і охоплює оцінку ризиків, управління взаємовідносинами з постачальниками, управління вразливостями та якістю продукції.
3) Належна практика повинна охоплювати всі різні суб’єкти, які відіграють певну роль у ланцюжку постачання продуктів і послуг ІКТ/ОТ, від виробництва до споживання.

4) Не всі сектори демонструють однакові можливості щодо управління ланцюжком постачання ІКТ/ОТ.

5) Необхідно додатково вивчити взаємозв’язок між директивою NIS2 і пропозицією щодо закону про кіберстійкість чи іншого законодавчого акту, галузевого чи негалузевого, який передбачає вимоги до кібербезпеки для продуктів і послуг.

У складному середовищі ланцюжків постачання впровадження належної практики кібербезпеки ланцюжка постачання на рівні ЄС зараз є важливішим, ніж будь-коли.

Директива NIS2 покращує кібербезпеку ланцюжка постачання наступним шляхом та забезпечує:

– усунення різниці між операторами основних послуг і постачальниками цифрових послуг;

– розширення охоплення більшої частини економіки та суспільства шляхом додавання більшої кількості секторів розмежування основних і важливих сутностей;

– звернення до кібербезпеки ланцюжка постачання та відносин з постачальниками, вимагаючи від окремих організацій розглядати відповідні ризики кібербезпеки;

– запровадження цілеспрямованих заходів, включаючи реагування на інциденти та управління кризовими ситуаціями, обробку та розкриття вразливостей, тестування кібербезпеки та ефективне використання шифрування;

– запровадження відповідальності керівництва кожного суб’єкта за дотримання заходів з управління ризиками кібербезпеки;

– запропонування групі співпраці з NIS проводити скоординовану оцінку ризиків безпеки конкретних критично важливих послуг, систем або продуктів у сфері інформаційно-комунікаційних технологій.

NIS2 Directive вимагає від основних і важливих організацій усунути ризики кібербезпеки в ланцюжках постачання та відносинах з постачальниками. Він робить це, вимагаючи в статті 21 основних і важливих суб’єктів вживання належних і пропорційних технічних, операційних та організаційних заходів з управління ризиками кібербезпеки та дотримуватися підходу до всіх небезпек. Ці заходи мають стосуватися, серед інших сфер, безпеки ланцюжка постачання, включаючи аспекти, пов’язані з безпекою, що стосуються відносин між кожним

суб’єктом господарювання та його безпосередніми постачальниками або провайдерами послуг.

Крім того, суб’єкти господарювання повинні брати до уваги вразливі місця, характерні для кожного прямого постачальника та провайдера послуг, а також загальну якість продуктів і практик кібербезпеки їхніх постачальників та провайдерів послуг, у тому числі їхні безпечні процедури розробки.

Основні та важливі організації зазвичай керують критичною інфраструктурою та використовують продукти, системи та рішення від виробників, постачальників каналів розподілу, системних інтеграторів і постачальників цифрових послуг. Деякі суб’єкти виробляють власні продукти (апаратне та програмне забезпечення) і в цьому випадку також можуть розглядатися як важливі суб’єкти господарювання. Для таких організацій також можна застосовувати рекомендовані належні практики для виробництва.

Суб’єкт господарювання зазвичай має договірні відносини зі своїми безпосередніми постачальниками та провайдерами послуг, де організаційні, технологічні та технічні заходи можуть бути визначені для відповідних постачань або придбаних послуг. Діапазон договірних узгоджених заходів обмежується повноваженнями організації щодо закупівель і можливостями постачальника чи провайдера послуг.

Деякі заходи розміщуються каскадом уздовж ланцюжка постачання, але загальний контроль за впровадженням відповідною організацією зазвичай неможливий, оскільки немає загальних договірних відносин, які могли б, наприклад, надавати право на аудит або право вимагати детальну інформацію про заходи безпеки від усіх постачальників по ланцюжку постачання.

Типовим прикладом такої відсутності контролю в ланцюжку постачання продуктів і компонентів є програмне забезпечення з відкритим вихідним кодом, яке є загальнодоступним, і правила використання якого визначаються в ліцензійних угодах, що не підлягають обговоренню. Іншим прикладом необхідності збереження контролю є закупівля послуг від постачальника послуг хмарних обчислень, оскільки це вимагає додаткових зусиль для забезпечення дотримання вимог General Data Protection Regulation.

Кібербезпека ланцюжка постачання ІКТ/ОТ відповідно до директиви NIS2 повинна розглядати такі аспекти в підході, що ґрунтується на оцінці ризику:

– управління ризиками ланцюжка постачання;

– відносини постачальників основних і важливих організацій з різними видами постачальників і постачальників послуг;

– усунення вразливостей у продуктах і компонентах;

– якість продукції та практики кібербезпеки постачальників і провайдерів послуг.

Ці аспекти можна структурувати в процесі безперервного вдосконалення «plan-do-check-act» (PDCA). Методологія добре визнана та відома з циклу покращення якості ISO 9001.

На рисунку 1.2 показано цикл PDCA для кібербезпеки ланцюжка постачання.

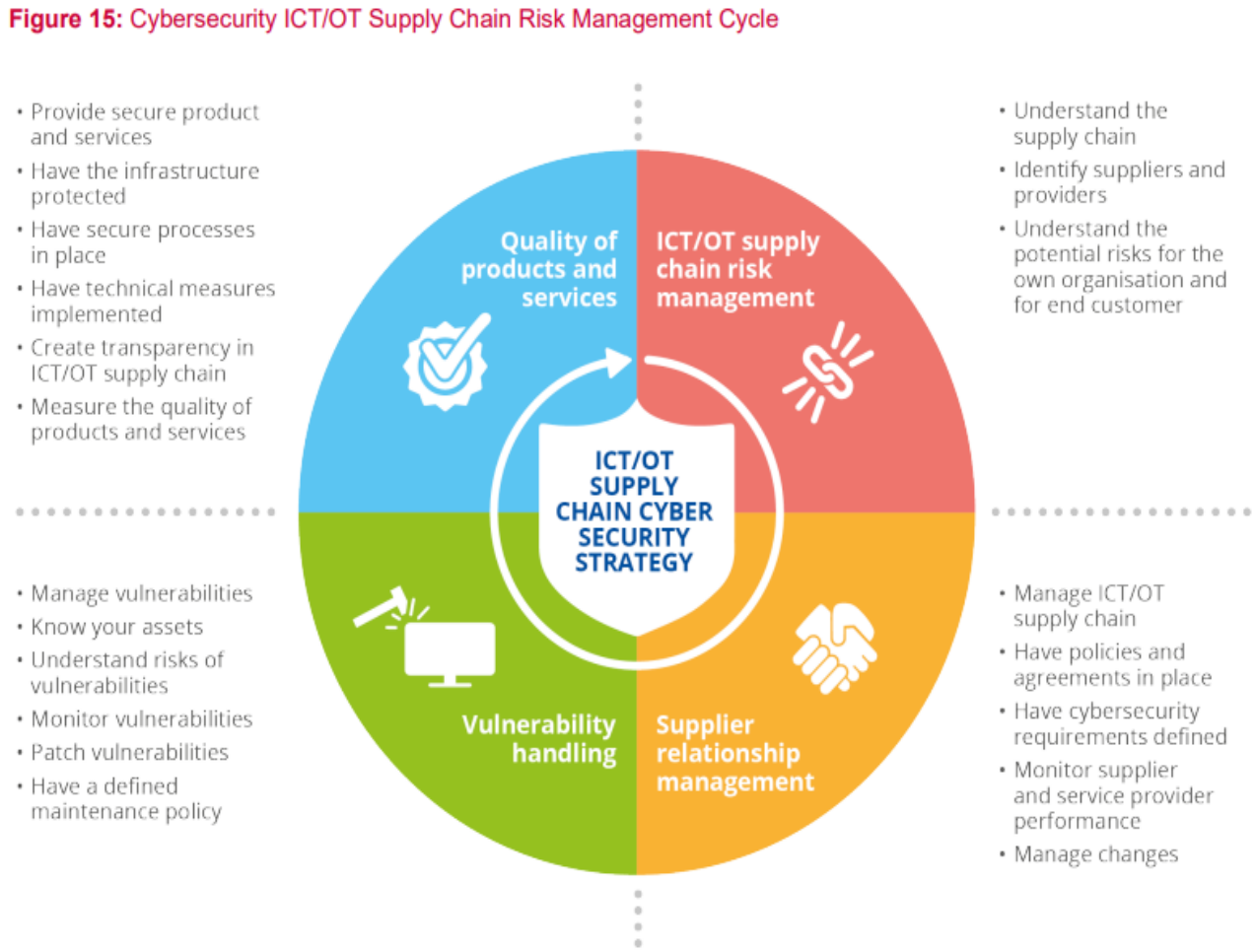

Рисунок 1.2 – Цикл PDCA для кібербезпеки ланцюжка постачання [4]

На рисунку 1.2 можна побачити цикл PDCA для кібербезпеки ланцюжка постачання. Він визначає наступні кроки.

Перший крок полягає в тому, що цикл починається з оцінки ризиків ланцюжка постачання ІКТ/ОТ, що допомагає організаціям зрозуміти відповідний ланцюжок постачання через ідентифікацію постачальників та провайдерів послуг, а також через розуміння потенційних ризиків ланцюжка постачання для власної організації та кінцевих споживачів, пов’язаних із постачанням від існуючих постачальників та провайдерів послуг.

Другий крок полягає в управлінні взаємовідносинами з постачальниками, що допомагає керувати ланцюжком постачання за допомогою політик, процедур і угод, спрямованих на ризики ланцюжка постачання. Це підтримується моніторингом ефективності постачальників та провайдерів послуг і практикою управління змінами.

Третій крок полягає в обробці вразливостей та визначає, як організація керує вразливими місцями.

Вразливості власних активів відстежуються та пов’язуються з активами в інфраструктурі. Їхні ризики розуміються, і розгортаються виправленнями, щоб закрити ці вразливості на основі чітко визначеної політики обслуговування.

Четвертим важливим фактором кібербезпеки ланцюжка постачання є якість продуктів і послуг, які використовуються в організації. Це вимагає, щоб учасники ланцюжка реалізовували процеси впровадження з практикою кібербезпеки, мали власну захищену інфраструктуру та технічні заходи в продуктах і послугах, що підвищить кібернадійність. Якість потрібно вимірювати та постійно вдосконалювати.

Основні та важливі суб’єкти потребують прозорості практики кібербезпеки продуктів і послуг, що надаються.

2 АНАЛІЗ АТАК НА ЛАНЦЮЖКИ ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ланцюжок постачання програмного забезпечення складається з будь-якого коду, компонентів, бінарних компонентів і інструментів, які беруть участь у життєвому циклі програми від розробки до виробництва. Він починається зі сховища або менеджера пакетів, будь-яких інструментів безперервної інтеграції CI, а також сценаріїв побудови та пакування, які дозволяють розгортати та запускати програму.

Ланцюжок постачання зазвичай включає такі етапи, як автоматизація збірки, контроль якості та тестування, а також автоматизацію розгортання.

Основною ціллю атак на ланцюжки постачання є постачальники і розробники програмного забезпечення. Основною метою таких атак є отримання доступу до вихідних кодів, створення або оновлення процесів шляхом зараження легальних програм для розповсюдження шкідливого програмного забезпечення.

2.1 Статистика атак на ланцюжок постачання програмного забезпечення

Факти про атаки на ланцюжки постачання програмного забезпечення тривожні.

Приведемо кілька основних статистичних даних щодо атак на ланцюжки постачання програмного забезпечення.

1) Argon, компанія Aqua Security, виявила, що кількість атак на ланцюжок постачання програмного забезпечення зросла більш ніж на 300 % у 2021 році.
2) Gartner прогнозує, що до 2025 року 45 % організацій зазнають атаки на ланцюжки постачання програмного забезпечення.
3) FBI повідомило про збільшення атак програм-вимагачів на 62 % з 2020 по 2021 рік.
4) Опитування Cloudbees показало, що 45 % підприємств визнали, що вони забезпечили захист лише половини свого ланцюжка постачання програмного забезпечення.

У звіті Gartner 2023 Supply Chain Risk Management Survey Report зазначається, що кількість атак на ланцюжки постачання зростає. 63 %

респондентів повідомили, що їхні організації зазнали атак на ланцюжки постачання за останній рік.

Це чітко вказує на занепокоєння, яке організація повинна мати щодо елементів свого ланцюжка постачання.

Наведені статистичні дані свідчать про те, що безпека ланцюжка постачання програмного забезпечення стане ще важливішою в найближчі роки, оскільки кількість атак даного типу зростає.

2.2 Останні відомі атаки на ланцюжки постачання

Атака SolarWinds, без сумніву, стала ключовим моментом, який підштовхнув світ технологій до усвідомлення небезпеки, що надходить від атак на ланцюжки постачання програмного забезпечення.

SolarWinds – це провідний інструмент моніторингу продуктивності мережі Network Performance Monitor (NPM), який використовується організаціями будь-якого розміру, включаючи державні установи.

У зловмисників був доступ до ланцюжка постачання SolarWinds більше року, перш ніж цей інцидент було фактично виявлено. Кожна клієнтська організація SolarWinds, у свою чергу, була скомпрометована. Ця подія була настільки масовою, що неможливо було легко визначити конкретну кількість атакованих клієнтів.

Через рік після того стався витік вихідного коду компанії Mercedes-Benz. Це сталося, у травні 2020 року, коли зловмисник зміг зареєструвати обліковий запис на порталі розміщення коду, а потім завантажив через Gitlab 580 сховищ Git, що містять вихідний код бортових логічних блоків (OLUs), встановлених у фургонах Mercedes.

Витік вихідного коду компанії Mercedes-Benz стався через відсутність процесів авторизації облікового запису. Причина витоку полягала в тому, що паролі та маркери API внутрішніх систем Daimler були розкриті, що погіршило ситуацію. Зловмисники могли використовувати паролі та ключі для здійснення майбутніх вторгнень у хмару та внутрішню мережу Mercedes-Benz.

Наступна атака CodeCov сталася у квітні 2021 року, коли сценарій CodeCov Bash Uploader було зламано та змінено.

CodeCov – це платформа для аналізу покриття коду та відстеження змін у тестах. Bash Uploader дозволяє клієнтам CodeCov передавати звіти про покриття коду для аналізу.

В ході атаки на CodeCov зловмисник використав образ Docker, який використовувався в ланцюжку постачання CodeCov, щоб отримати доступ. Це призвело до того, що всі клієнти CodeCov також стали вразливими до атаки, оскільки CodeCov – це інструмент, що вбудований у ланцюжок постачання програмного забезпечення їхніх клієнтів.

У лютому 2022 року сталася атака на Viasat. Після початку російської агресії проти України в лютому 2022 року, атака на мережу Viasat KA-SAT призвела до перебоїв у споживацькій супутниковій широкосмуговій мережі. Збій в Інтернеті торкнувся майже 9 тисяч абонентів Nordnet, яка є клієнтом Viasat у Франції. Атака мала побічний вплив на додаткових 40 000 абонентів Eutelsat, які покладалися на супутникову мережу Viasat KA-SAT. Збої спостерігалися в Німеччині, Греції, Франції, Італії, Угорщині та Польщі і тривали кілька днів. Viasat повідомила, що це не вплинуло на мобільність, що безпосередньо керується, або на державних користувачів.

Атака на репозиторій Node Package Manager сталася у березні 2022 року. Зловмисник під назвою «RED-LILI» був пов’язаний із великомасштабною кампанією атаки на ланцюжок постачання, націленою на репозиторій Node Package Manager. В результаті даної атаки було опубліковано близько 800 шкідливих модулів.

У січні 2023 року компанія Airbus також постраждала від атаки на ланцюжок постачання, що була здійснена суб'єктом загрози, відомим як USDoD.

Компанія Airbus підтвердила, що атака була здійснена через скомпрометований обліковий запис співробітника Turkish Airlines, одного з клієнтів Airbus. В результаті зловмисник зміг отримати доступ до облікового запису співробітника і отримати доступ до систем Airbus.

Порушені дані включали особисту інформацію, пов'язану з більш ніж 3 000 постачальників Airbus, таких як Rockwell Collins і Thales Group. Дамп даних включав імена, адреси, номери телефонів і адреси електронної пошти.

У лютому 2023 року сталася атака на ланцюжок постачання в Каліфорнійському університеті у Сан-Франциско (UCSF). В результаті система електронних медичних записів (EHR) була недоступна протягом декількох днів. Без доступу до системи EHR лікарі UCSF не могли отримати доступ до медичних

карток пацієнтів або запланувати операції. Це призвело до скасування або перенесення кількох операцій.

Зловмисники здійснили атаку, скориставшись вразливістю в Codecov, популярному програмному забезпеченні для тестування коду, яке компанія Zellis, розробник програмного забезпечення для клінічних досліджень, використовує для перевірки свого програмного забезпечення на вразливості. UCSF використовував Zellis та, в результаті, постраждав від атаки.

Зловмисники змогли викрасти особисту інформацію учасників клінічних випробувань з систем Zellis. Частина даної інформації була опублікована в Інтернеті.

Також у 2023 році зловмисниками було проведено атаку на ланцюжок поставок компанії Norton. Ця компанія пропонує продукти та послуги, що допомагають забезпечити цифрову безпеку, захист особистих даних та конфіденційність в Інтернеті. Найбільш відомим програмним забезпеченням є антивірус Norton Antivirus, який є широко використовуваним антивірусним програмним забезпеченням. Але вони також постраждали від атаки на ланцюжок постачання у травні 2023 року.

Дана атака використовувала вразливість нульового дня в MOVEit Transfer, програмному забезпеченні для керованої передачі файлів MFT (Master File Table), яке материнська компанія Norton, Gen Digital, використовує для передачі файлів між своїми офісами та клієнтами.

Зловмисники змогли отримати доступ до мережі Norton і викрасти особисту інформацію співробітників, включаючи імена, адреси, дати народження та робочі адреси електронної пошти. Зловмисники також погрожували оприлюднити викрадені дані, якщо компанія Norton не заплатить викуп.

У березні 2023 року Colonial Pipeline також постраждав від атаки на ланцюжок постачання. Це найбільша трубопровідна система для переробки нафтопродуктів у США.

Атака використовувала вразливість віддаленого виконання коду Remote Code Execution (RCE) в PulseConnect Secure, програмному забезпеченні VPN, яке використовується Colonial Pipeline для моніторингу роботи трубопроводу. Зловмисники змогли отримати доступ до мережі Colonial Pipeline і зашифрувати її системи.

Атака унеможливила експлуатацію трубопроводу Colonial Pipeline. В результаті робота нафтопроводу була припинена на п'ять днів. Це спричинило

дефіцит бензину на південному сході США. Colonial Pipeline заплатила зловмисникам викуп у розмірі 4,4 мільйона доларів, щоб відновити доступ до своїх систем.

У лютому 2023 року компанія Microsoft також постраждала від атаки на ланцюжок постачання програмного забезпечення.

Атака використовувала вразливість в Jfrog Artifactory, менеджері бінарних репозиторіїв, який Microsoft використовує для зберігання та розповсюдження своїх програмних компонентів. Зловмисники змогли отримати доступ до Jfrog Artifactory та впровадити шкідливий код у деякі програмні компоненти Microsoft. Це дозволило зловмисникам отримати доступ до мереж Microsoft і викрасти вихідний код та іншу конфіденційну інформацію [15].

На рисунку 2.1 показано перелік основних атак на ланцюжки постачання, що був опублікований Національним центром контррозвідки та безпеки США (NCSC).

Legend	Discovered	Incident	Entry Point	Compromised Stage	Affected Software	Initial Impact	Notes
1	Feb 2021	Birsan research (Ethical hacker)	Open-Source Libraries	Development (open-source library)	Multiple	Proof-of-concept	Security researcher Alex Birsan identified improperly configured package managers at multiple major companies and verified they would install unauthorized code from public repositories instead of limiting access to internal servers.
2	Dec 2020	VGCA compromise (SignSight)	Government Certification Authority Website	Deployment (infrastructure)	Digital Signature Toolkit	Targeted government and commercial entities	Compromised a Vietnam government certificate authority and added a backdoor component to installers for legitimate software.
3	Dec 2020	SolarWinds Orion compromise	Undisclosed	Development (infrastructure)	Network Monitoring and Management Platform	Espionage	The SolarWinds Orion source code compromise represents the most significant cyber incident impacting enterprise networks across the private sector, federal, state, and local governments to date.
4	Nov 2020	VeraPort compromise	Compromised Website (Watering Hole)	Deployment (digital certificates)	Computer Utility (Browser Plugin)	Targeted government and financial websites	Targeted South Korean users of a trusted download verification tool by prompting its browser plugin to install malware signed with stolen authentic digital certificates.
5	Jul 2020	Twilio SDK compromise	Misconfigured Public Cloud Storage Bucket	Development (SDK tool)	Cloud-Based Communications	Theft	Attackers injected malicious code within the SDK library of a Communications Platform as a Service (CPAAS) company through its misconfigured cloud-hosted infrastructure.
6	Jun 2020	GoldenSpy (MITRE ID: S0493)	Over Distribution with Hidden Malicious Properties	Design (intentional)	Business Software	Targeted specific Western companies	A Chinese bank compelled Western corporate clients to install tax software containing a hidden backdoor.
7	Jan 2019	Asus compromise (ShadowHammer)	Compromised Development Infrastructure	Development (digital certificates)	Computer Utility (Software Updater)	Targeted specific individuals	Compromised manufacturer to target a pool of specific customers by delivering malware via software updates signed with authentic certificates.
8	Nov 2018	Copay compromise	Open-Source Library	Development (open-source code)	Cryptocurrency Wallet	Cryptocurrency theft	Poisoned popular open-source JavaScript library by injecting malicious code to steal cryptocurrency stored in desktop and mobile wallet software.
9	Aug 2018	AppleJeus campaign	Overt Distribution with Hidden Malicious Properties	Design (intentional)	Cryptocurrency Apps	Cryptocurrency theft	Overt distribution of software with hidden malicious properties. Persistent campaign developed and distributed innocent-looking cryptocurrency applications that contained hidden malicious content.
10	Jun 2017	NotPetya (MITRE ID: S0368)	Compromised Software Update Infrastructure	Deployment (infrastructure)	Business Software	Data destruction; disrupted commerce and services	Self-propagating data-destruction malware delivered through a software update from the developer's compromised infrastructure.

Рисунок 2.1 – Перелік основних атак на ланцюжки постачання [19]

На рисунку 2.1 можна побачити перелік основних атак на ланцюжки постачання, опублікований Національним центром контррозвідки та безпеки США (NCSC) 21 березня 2021 року. На рисунку представлено дату виявлення та назву інциденту, точку входу, скомпрометований етап, уражене програмне забезпечення, початковий вплив та опис проведених атак.

NCSC було опубліковано наступні інциденти: Birsan research (Ethical hacker), VGCA compromise (SignSight), SolarWinds Orion compromise, VeraPort compromise, Twilio SDK compromise, GoldenSpy, Asus compromise (ShadowHammer), Copay compromise, AppleJeus campaign, NotPetya. Дані атаки в основному було проведено

на стадії розробки (це торкнулося бібліотеки з відкритим кодом, інфраструктури, SDK tool, цифрових сертифікатів та відкритого вихідного коду), розгортання (це було пов’язано з інфраструктурою та цифровими сертифікатами) та дизайну (в даних випадках атаку було проведено навмисно організаціями – GoldenSpy та AppleJeus campaign).

В результаті проведення даних атак в числі ураженого програмного забезпечення виявилися: набір інструментів цифрового підпису, платформа моніторингу та керування мережею, комп’ютерні утиліти (плагін для браузера, програма оновлення програмного забезпечення), хмарні комунікації, програмне забезпечення для бізнесу, криптовалютний гаманець, програми для криптовалют. Деякі з атак уразили безліч програмного забезпечення.

Розглянемо типи атак на ланцюжки постачання програмного забезпечення.

Перший тип атак – порушення CI/CD Pipeline.

Зловмисники шукають способи проникнути в конвеєр CI/CD, який використовується організаціями для доставки програмного забезпечення. Оскільки конвеєр CI/CD є «центральною нервовою системою» всього процесу розробки програмного забезпечення, будь-які зміни, внесені до нього, негативно вплинуть на виробничі програми, а також програми клієнтів.

Другий тип – неправильні налаштування інструмента CI/CD.

Оскільки практики розробки програмного забезпечення модернізується, конфігурація як код є найкращою практикою. Це передбачає кодифікацію конфігурації таких аспектів, як інфраструктура та політики, які керують програмними процесами. Така конфігурація зберігається у вигляді файлів YAML. Часто контроль над цими конфігураційними файлами не захищений належним чином, залишаючи вектори атак і вразливості відкритими для зловмисників. У чужих руках ці конфігураційні файли можуть бути зловживані.

Третій тип – скомпрометовані інструменти для створення програмного забезпечення.

Є багато інструментів, які складають сучасний ланцюжок постачання програмного забезпечення, і список тільки збільшується з кожним днем. Дані інструменти варіюються від програмного забезпечення з відкритим кодом до комерційних інструментів. Ці інструменти виконують різні ролі, зокрема створюють збірки, перевіряють якість і розгортають код у виробництві. Важливо захистити ці інструменти та переконатися, що вони не стануть вектором впровадження шкідливого коду в конвеєр для загрозливого суб’єкта.

Четвертий тип – атака плутанини залежностей.

Менеджери пакетів є важливою частиною роботи з будь-якою мовою програмування, таких як Node.js або Python. Завантаження пакетів із цих платформ пов’язане з ризиком, оскільки будь-хто може завантажити пакет на них. Останнім часом зловмисники знайшли спосіб обманом змусити розробників завантажувати шкідливі пакети, націлюючись на орфографічні помилки в пакетах, що найчастіше завантажуються. Оскільки розробники здебільшого вводять назви пакетів в інтерфейсі командного рядка – це призводить до помилок.

П’ятий тип – загрози вихідного коду.

Незахищений код і скомпрометовані системи контролю вихідного коду є одними з найпоширеніших загроз вихідного коду. Людська халатність та відсутність безпечних методів кодування можуть створити вразливість даного типу. Крім того, незахищені робочі станції розробки можуть впроваджувати незахищений або шкідливий код.

Реалізація політики на робочих станціях розробників, регулярне сканування коду і API під час і після розробки, а також запровадження методів безпечного коду можуть допомогти пом’якшити загрози незахищеного коду. Однак також необхідно переконатися, що розробники не мають можливості додавати незахищений або зловмисний код у сховища вихідних кодів, шляхом включення перевірки змін вихідного коду.

Код і репозиторії вихідних кодів можуть залишатися в безпеці, лише якщо конкретна система контролю вихідного коду не буде скомпрометована. Забезпечити безпеку можна шляхом обмеження доступу до цієї системи та інших систем у конвеєрі збірки.

Можна використовувати багатофакторну автентифікацію (MFA), щоб захистити доступ до системи та регулярно оцінювати інтеграцію джерела, включаючи конфігурації та сценарії [2].

2.3 Методи зменшення ризиків безпеки

Незважаючи на труднощі, дотримання деяких практик для керування ланцюжком постачання програмного забезпечення, допоможе зменшити ризики безпеки.

По-перше, необхідно використовувати контрольні списки для контролю процесів.

Контрольні списки є простим і перевіреним способом забезпечення безпеки в масштабах. Кожна команда та кожен її учасник повинні мати власний контрольний список. Він може відрізнятися для кожної особи та команди, але це є потужним інструментом для забезпечення стандартів безпеки.

По-друге, потрібно зменшити поверхню атаки. Ця давня практика безпеки актуальна й сьогодні. Необхідно надати зловмисникові мало можливостей або не надавати жодних для здійснення атаки. Цього можна досягти шляхом видалення старих та невикористовуваних інструментів та компонентів з ланцюжка постачання, зберігаючи кодову базу додатків невеликою та легкою, а також шляхом скорочення компонентів інфраструктури до тих, що використовуються на даний час. Необхідно видалити непотрібних користувачів і обмежити їх права, виходячи з їх обов’язків та поставлених задач. Усе це сприяє підвищенню рівня безпеки системи.

По-трете, необхідно сканувати кожен крок ланцюжка постачання.

Оскільки кожен крок ланцюжка постачання є вразливим до атак, код слід перевіряти на кожному етапі процесу. Таке сканування не повинно виконуватися вручну жодною особою, скоріше, це робота спеціальних програмних інструментів сканування. Сканування має тривати безперервно та повідомляти про будь-які невідповідності, уразливості та порушення.

Важливо також переконатися, що партнерські програми та інтеграції є безпечними. Очікується, що партнери та постачальники дотримуватимуться однакового набору стандартів для конкретної організації. Інтеграції слід ретельно перевіряти та регулярно оновлювати, щоб вони не містили вразливостей. Необхідно регулярно перевіряти, як програми партнерів отримують доступ і використовують дані конкретної організації.

Необхідно використовувати безпеку та тестування на проникнення, а також сприяти культурі, в якій тестувальників заохочують щось зламувати та перевіряти межі системи. Можна стимулювати етичних хакерів шукати вразливості та навіть

створювати програми винагороди за зусилля з пошуку помилок. Все це дозволить бути на крок попереду зловмисників.

Слід переконатися, що все програмне забезпечення є оновленим до останніх версій. З огляду на численні пакети програмного забезпечення, які сьогодні використовуються в ланцюжку постачання, оновлення програмного забезпечення може бути повноцінною роботою як для спеціалістів із безпеки, так і для розробників. Будь-яка допомога, яку вони можуть отримати, щоб полегшити цю роботу, посилить безпеку системи.

Використання графіків залежностей також може зменшити ризики безпеки. Графіки залежностей є способом візуалізації того, як компоненти системи залежать один від одного. Вони корисні для відстеження впливу атаки та вжиття профілактичних заходів, щоб гарантувати, що кожна частина системи оновлена та сумісна з іншими частинами [3].

3 ІСНУЮЧІ МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ НА КОЖНОМУ ЕТАПІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглянемо основні вразливості безпеки ланцюжка постачання.

Першою вразливістю є відсутність видимості.

Для ланцюжків постачання програмного забезпечення моніторинг є великою проблемою, оскільки різні люди керують різними частинами ланцюжка, а команди та інструменти відрізняються.

Для забезпечення безпеки необхідно зібрати всі дані з кожного кроку та систематизувати їх у єдиному місці, хоча це є нелегкою задачею.

Друга вразливість – схильні до помилок ручні процеси.

Автоматизація процесів є ключовим принципом CI/CD. Однак насправді CI/CD працює лише напівавтоматично з великою кількістю ручного втручання на кожному кроці.

Організації, які йдуть на компроміс щодо автоматизації та погоджуються на ручні операції, стикаються з уразливістю, причиною якої є людська халатність або зловмисники усередині організації. Навіть для організацій, яким вдалося повністю автоматизувати процеси свого ланцюжка постачання, ризик полягає у встановленні засобів контролю та перевірок безпеки. Без цього автоматизація може бути шкідливою.

Третя вразливість полягає у складності ланцюжка постачання. Потрібні інтеграції були створені завдяки безперебійному конвеєру CI/CD за допомогою різних найкращих у своєму класі інструментів сторонніх розробників. Вони зобов’язані ввімкнути спеціальні робочі процеси за участю сторонніх постачальників і партнерів. Ці інтеграції є живильним середовищем для вразливостей і легко підбираються зловмисниками.

Наступною вразливістю є неправильне поводження з секретами.

Конфіденційна інформація має багато форм, таких як паролі, маркери, ключі шифрування та хеші. Ця секретна інформація не може бути жорстко закодована в програмі або зберігатися в незашифрованих файлах. Їх потрібно обробляти спеціально створеними інструментами керування секретами. Проте вивчення цього нового способу обробки секретів не є пріоритетом для команд Devops, що призводить до порушення безпеки [5].

Відсутність практик безпеки API (Application Programming Interface) також є однією з вразливостей безпеки ланцюжка постачання.

Інтерфейс прикладного програмування (API) – це така складова, що об'єднує хмарні системи. Вони є шлюзом сторонніх систем для доступу до послуг організації. Якщо вони скомпрометовані, зловмисники можуть легко отримати доступ до більш глибоких частин системи.

Існують також уразливості у відкритому вихідному коді. Найновішою та найвідомішою з відкритих вразливостей є Log4j. Було багато таких випадків, коли розробники нехтували безпекою інструментів з відкритим кодом. На такі проекти не було фінансування, тому не дивно, що через деякий час вони переставали вживатися. Відповідальність за забезпечення безпеки повинна лежати на компанії, яка використовує ці інструменти з відкритим кодом.

Небезпечним є вразливе або невиправлене програмне забезпечення.

Системи IoT і навіть багато застарілих систем програмного забезпечення не обслуговуються активно, а їх мікропрограмне або програмне забезпечення застаріло. Продовжувати перевіряти наявність застарілого програмного забезпечення та видаляти його з потрібної системи – складно, але це необхідно, якщо метою є «герметична» система від початку до кінця [3].

Для захисту від атак на ланцюжки постачання програмного забезпечення можна також використовувати програмне забезпечення Aqua. Дане програмне забезпечення є дієвим рішенням для забезпечення безпеки ланцюжка постачання.

Aqua сканує кожен крок конвеєра CI/CD, шукаючи вразливості, і повідомляє про будь-які аномалії, що були знайдені в процесі сканування. Завдяки готовій інтеграції для найпопулярніших інструментів CI/CD, таких як GitHub, GitLab, Jenkins тощо, програмне забезпечення Aqua допоможе, незалежно від того, які інструменти CI/CD використовуються.

ПЗ Aqua охоплює весь ланцюжок постачання і консолідує всі дані моніторингу в одному місці, а також надсилає сповіщення про них у режимі реального часу. Все це є потужним і дієвим способом боротьби із загрозами безпеки конкретного ланцюжка постачання програмного забезпечення.

Тому програмне забезпечення Aqua можна використовувати для забезпечення глибокої видимості та безпеки ланцюжка постачання програмного забезпечення.

3.1 Життєвий цикл безпечної розробки програмного забезпечення (SSDLC)

Розглянемо, що представляє собою безпечний цикл розробки програмного забезпечення – Secure Software Development Life Cycle (SSDLC).

Secure SDLC вимагає додавання тестування безпеки на кожному етапі розробки програмного забезпечення, від проектування до розробки, розгортання та в подальшому. Приклади включають розробку додатків для забезпечення безпеки архітектури, а також включення факторів ризику безпеки на початковому етапі планування.

Безпека є необхідною частиною будь-якої програми, яка включає важливі функції.

Безпека повинна забезпечуватися на кожному етапі життєвого циклу розробки програмного забезпечення (SDLC) і має бути в центрі уваги розробників, коли вони виконують вимоги до програмного забезпечення організації.

Розглянемо способи створення Secure SDLC, який допоможе виявити проблеми у вимогах до того, як вони виявляться проблемами безпеки у виробництві.

Завдяки цілеспрямованим зусиллям і правильним рішенням безпеки проблеми безпеки можна вирішити в конвеєрі SDLC задовго до розгортання в робочій мережі. Це зменшить ризик виявлення вразливостей безпеки у конкретній програмі та мінімізує вплив у разі їх виявлення.

Мета Secure SDLC полягає не в повній ліквідації традиційних перевірок безпеки, таких як тести на проникнення, а в тому, щоб включити безпеку в сферу обов’язків розробників та надати їм можливість створювати безпечні програми з самого початку.

Secure SDLC є необхідним, тому що безпека програми є вкрай важливою. Розробники повинні знати про потенційні проблеми безпеки на кожному кроці процесу. Це вимагає інтеграції безпеки у SDLC способами, що раніше не вважалися необхідними.

Оскільки будь-хто потенційно може отримати доступ до вихідного коду, необхідно переконатися, що кодування виконується з урахуванням потенційних вразливостей.

Таким чином, наявність надійного та безпечного процесу SDLC має вирішальне значення для того, щоб конкретна програма не піддавалася атакам зловмисників та інших нечесних користувачів.

Нові інструменти, такі як керування безпекою додатків, можуть допомогти забезпечити цілісне уявлення про компоненти налаштування безпеки додатків, а також надати контекст про існуючі вразливості.

Життєвий цикл розробки програмного забезпечення (SDLC) представлений на рисунку 3.1. Він описує, яким чином відбувається створення програмних додатків.

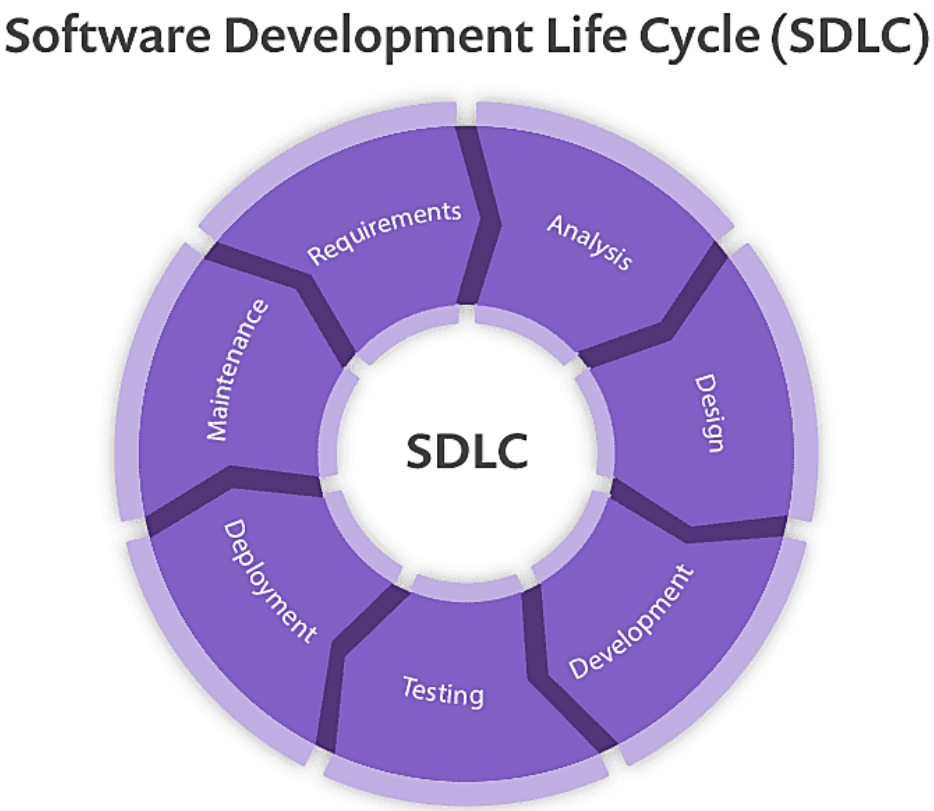

Рисунок 3.1 – Життєвий цикл розробки програмного забезпечення (SDLC) [20]

На рисунку 3.1 можна побачити життєвий цикл розробки програмного забезпечення (SDLC).

Хоча розробку програмного забезпечення часто вважають простим написанням коду, насправді існує кілька етапів життєвого циклу розробки програмного забезпечення до доставки, де програмування є лише одним з них. Ці етапи включають збір вимог, аналіз, проектування, розробку, тестування, розгортання та обслуговування.

Фази SDLC не обов’язково є лінійною послідовністю, залежно від використовуваних методологій SDLC. Етапи можуть фактично накладатися або змінювати порядок.

Як показано на рисунку 3.1, зазвичай SDLC містить наступні етапи:

– збір вимог;
– аналіз вимог керівництва до проектування;
– розробка нових функцій на основі вимог;
– розвиток нових можливостей (написання коду відповідно до вимог);
– тестування та перевірка нових можливостей – підтвердження того, що надані можливості справді відповідають вимогам;
– розгортання нового проекту;
– підтримка та розвиток цих можливостей після випуску.

Фази безпечного життєвого циклу розробки програмного забезпечення SSDLC представлені на рисунку 3.2.

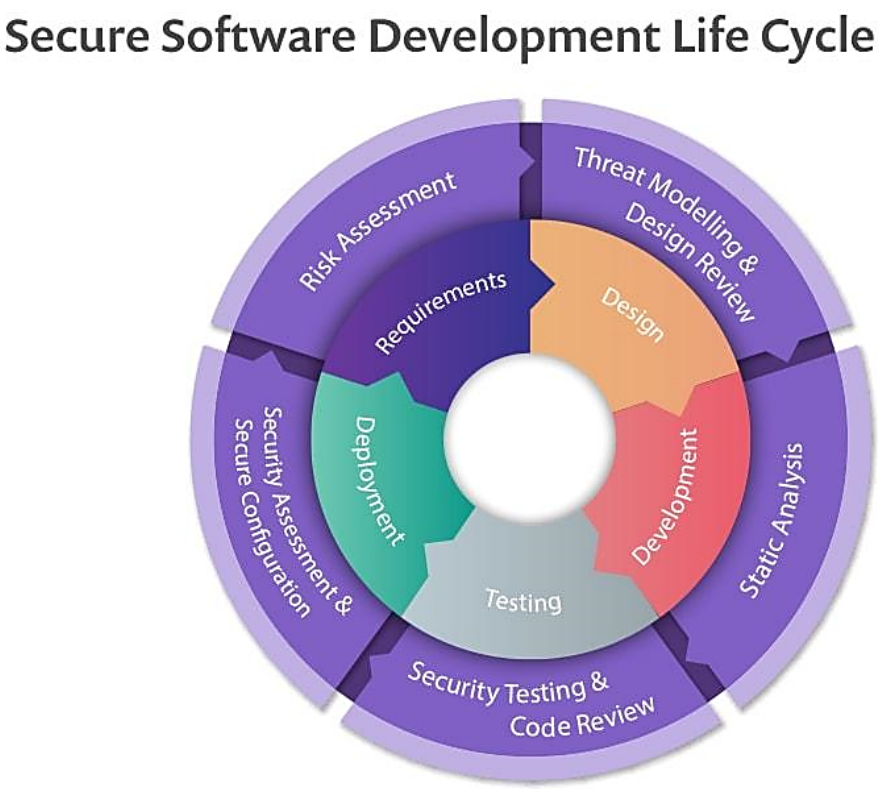

Рисунок 3.2 – Безпечний життєвий цикл розробки програмного забезпечення (SSDLC) [20]

На рисунку 3.2 можна побачити життєвий цикл безпечної розробки програмного забезпечення (SSDLC).

Кожна фаза SDLC має сприяти безпеці програми в цілому. Це досягається різними способами для кожної фази SDLC. Але одним критичним зауваженням є те, що безпека життєвого циклу розробки програмного забезпечення має бути в центрі уваги всієї команди.

Розглянемо фази безпечного життєвого циклу розробки програмного забезпечення.

Першою фазою SSDLC є вимоги. На цьому ранньому етапі запити щодо нових функцій збираються від різних зацікавлених сторін. Важливо визначити будь-які міркування безпеки щодо функціональних вимог, які збираються для нового випуску.

Друга фаза – дизайн. Ця фаза перетворює вимоги до сфери застосування на план того, як це має виглядати у фактичній заявці. Тут функціональні вимоги зазвичай описують те, що має статися, тоді як вимоги безпеки зазвичай зосереджуються на тому, чого статися не повинно.

Третьою фазою є розвиток. Коли настає час фактично реалізувати дизайн і втілити його в життя, проблеми зазвичай переходять на те, щоб код був добре написаний з точки зору безпеки. Зазвичай існують встановлені вказівки щодо безпечного кодування, а також перевірки коду, які двічі перевіряють правильність дотримання цих вказівок. Ці перевірки коду можуть бути ручними або автоматизованими за допомогою таких технологій, як статичне тестування безпеки додатків SAST.

Тим не менш, сучасні розробники додатків не можуть турбуватися лише про код, який вони пишуть, оскільки, як відомо, переважна більшість сучасних додатків не пишеться з нуля. Натомість розробники покладаються на існуючу функціональність, яка зазвичай надається безкоштовними компонентами з відкритим кодом, щоб якнайшвидше надати нові функції. Насправді понад 90 % сучасних розгорнутих програм створено з цих компонентів з відкритим кодом. Такі компоненти з відкритим кодом зазвичай перевіряються за допомогою інструментів аналізу складу програмного забезпечення – Software Composition Analysis (SCA).

Інструкції щодо безпечного кодування в цьому випадку можуть включати:

– використання параметризованих SQL-запитів лише для читання для зчитування даних із бази даних і мінімізації ймовірності того, що будь-хто може коли-небудь використати ці запити для зловмисних цілей;

– перевірку введених користувачем даних перед обробкою даних, що містяться в них;

– очищення будь-яких даних, які повертаються користувачеві з бази даних;

– перевірку відкритих бібліотек на наявність вразливостей перед їх використанням.

П’ята фаза представляє собою перевірку.

На етапі перевірки програми проходять ретельний цикл тестування, щоб переконатися, що вони відповідають початковому дизайну та вимогам. Це також є чудовим місцем для впровадження автоматизованого тестування безпеки за допомогою різних технологій. Програма не буде розгорнута, якщо такі тести не було пройдено. Ця фаза часто включає автоматизовані інструменти, такі як конвеєри CI/CD для контролю перевірки та випуску.

Перевірка на цьому етапі може включати:

– автоматичні тести, які виражають критичні шляхи конкретної програми;

– автоматизоване виконання модульних тестів програми, які перевіряють правильність основної програми;

– інструменти автоматизованого розгортання, які динамічно змінюють секрети програми для використання у виробничому середовищі.

Шоста фаза – технічне обслуговування та розвиток.

Після того, як програму було випущено, забезпечення її безпеки не повинно припинятися. Фактично, уразливості, які не були виявленні на попередніх етапах, можуть бути знайдені в програмі через довгий час після її випуску. Такі вразливості можуть міститися в коді, що був написаний розробниками, але все частіше їх виявляють в базових компонентах з відкритим кодом, які є складовими програми. Це призводить до збільшення кількості «нульових днів» – раніше невідомих вразливостей, що виявляють у виробництві розробники програми.

Наступним кроком команда розробників має виправити виявлені вразливості. Цей процес у деяких випадках може вимагати значного переписування функціональності програми. Уразливості на даній стадії також

можуть походити з інших джерел, наприклад, зовнішніх тестів на проникнення, що проводяться етичними хакерами, або поданням від громадськості через так звані програми «bug bounty». Вирішення таких типів виробничих проблем має бути заплановано та враховано в майбутніх випусках.

Забезпечення Secure SDLC вимагає зосередження не на тому, як працює програма, а на тому, як розробники перетворюють вимоги в код програми. Зрозуміло, що під час розробки програми, її безпека має бути на першому місці. Це може вимагати культурних змін у командах та змін автоматизованих процесів, а також перевірок на кожному етапі розробки програмного забезпечення.

Забезпечення SSDLC для програми значною мірою залежить від сильних і слабких сторін команди розробників програмного забезпечення, що працює над безпекою SDLC, тому важко визначити єдиний безпечний процес SDLC.

Оскільки Secure SDLC передбачає зміну існуючих процесів, впровадження нових інструментів і, що більш важливо, стимулювання культурних змін у ряді команд, шлях до добре функціонуючого SSDLC зазвичай унікальний для кожної організації та може навіть відрізнятися в різних бізнес-підрозділах.

Розглянемо найкращі методи безпеки SDLC.

По-перше, необхідно проводити навчання своїх розробників.

Secure SDLC пов'язаний з багатьма ініціативами, зокрема: створення правил безпечного кодування; надання розробникам інформації про безпеку та навчання безпечному кодуванню; встановлення чітких очікувань щодо того, як швидко потрібно вирішувати проблеми, виявлені у виробництві (також відомі як угоди про рівень обслуговування).

По-друге, потрібно мати чіткі вимоги. Тому, щоб не було створено, це має бути легким для розуміння. Командам розробників потрібні чіткі вимоги, які буде легко виконувати. Це стосується всіх порад, рекомендацій і вказівок щодо безпеки. Будь-яка вразливість, виявлена під час тестування, має бути відносно легкою для усунення. Важливо, щоб усі задіяні люди, процеси та інструменти виносили рішення на стіл, а не просто вказували на проблеми.

Наступним – важливо підтримувати зростання мислення. Оскільки SSDLC змінить те, як працюють і взаємодіють кілька команд, важливо, щоб кожен ставився до цього досвіду з відкритим розумом, а для команди безпеки було проведено налаштування, завдяки яким розробникам буде надана можливість захищати власні програми.

Також потрібно пов’язати реалізацію з іншими ініціативами. Для добре налагоджених додатків і команд часто може бути легше впровадити зміни SSDLC, якщо це пов’язано з іншими зусиллями з модернізації, такими як хмарна трансформація, ініціатива DevOps або її варіація з більшою безпекою DevSecOps.

Також важливо зазначити, шо спершу необхідно вирішувати великі проблеми. Тобто, необхідно зосередитися на найважливіших проблемах і ефективних виправленнях, а не на вирішенні кожної знайденої вразливості. Незважаючи на те, що у нових або менших додатках є можливість виправлення кожної проблеми безпеки, яка існує, це не обов’язково працюватиме в старіших і більших додатках. Тому підхід сортування також може бути корисним. Це зосереджено не лише на тому, щоб запобігти виникненню проблем із безпекою, а й на тому, щоб із часом перевірити наявні вразливості й усунути їх.

Важливо обговорити зв’язок між SSDLC і DevSecOps. Іноді вони використовуються як синоніми, що може призвести до плутанини. Хоча SSDLC і DevSecOps тісно пов’язані між собою, вони фактично доповнюють один одного. Як SSDLC, так і DevSecOps зосереджуються на тому, щоб розробники могли мати більше контролю над своїми програмами, гарантуючи, що вони роблять більше, ніж просто пишуть і тестують свій код відповідно до функціональних специфікацій.

Secure SDLC зосереджений на тому, як розроблено та створено додаток. DevSecOps прагне перенести власність на виробниче середовище для кожної програми від традиційних IT-команд до рук розробників. Це дозволяє розробникам максимально зосередитися на автоматизації процесів збирання, тестування та випуску.

DevOps і DevSecOps почали революцію в переосмисленні ролі розробників програмного забезпечення. Цьому, звичайно, сприяли інші серйозні зміни, такі як хмарні перетворення. Але хоча розширення можливостей розробників і прискорення тестування безпеки є ключовим фактором успіху для більшості сучасних організацій, було б помилкою розглядати безпеку додатків лише як проблему автоматизації. Натомість важливо впроваджувати культурні та технологічні зміни, які сприятимуть підвищенню обізнаності та міркувань безпеки на ранніх стадіях процесу розробки. Це повинно пронизувати всі частини життєвого циклу розробки програмного забезпечення, незалежно від того, чи це називається SSDLC, чи DevSecOps.

Традиційних методів тестування на вразливості у виробництві вже недостатньо для захисту конкретних програм. Однією з причин цього стало те, що з розвитком індустрії програмного забезпечення змінювалися й типи атак. Саме тому розгортання та підтримка безпечної програми вимагає захисту кожного етапу процесу розробки програми. Це означає, що необхідно задавати питання про поведінку безпеки на етапі збору вимог, коригувати командну культуру та практики для врахування орієнтованого на безпеку мислення, впроваджувати автоматизовану перевірку в процес розгортання та багато інших практик, які разом створюють безпечний процес SDLC.

SSDLC дозволяє змістити ризики безпеки вліво, вирішуючи джерело проблем безпеки на етапі вимог замість того, щоб повертатися до цього з етапу обслуговування. Зосереджуючись на забезпечені безпеки на кожному етапі розробки, можна бути впевненими, що в результаті програма буде набагато захищеною.

4 СТВОРЕННЯ МОДЕЛІ ЗАХИСТУ ЛАНЦЮЖКІВ ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В епоху цифровізації забезпечення ефективних практик кібербезпеки для кожної організації є абсолютною необхідністю. Важливим є не тільки захист інфраструктури в штаб-квартирі компанії, а й забезпечення безпеки всього ланцюжка постачання. Через те, що компанії та ринки у глобальному масштабі взаємопов’язані, атака на один суб'єкт може мати міжнародні негативні наслідки.

Атака на ланцюжок постачання – це не що інше, як атака на одного з партнерів – компанії та організації, з якими співпрацює умовне підприємство, доступ до інформаційних систем яких кіберзлочинці можуть отримати через прогалини у безпеці.

Згідно зі звітом IBM за 2020 рік, уразливості програмного забезпечення відповідають за 16 % таких інцидентів. Крім того, такий інцидент може обернутися кризою глобального масштабу.

Ланцюжки постачання можуть мати величезне охоплення та дуже складну мережу зв’язків, тому виявити такі атаки надзвичайно важко. Без суворої політики безпеки та постійного створення потужного захисту вся бізнес-мережа може опинитися під загрозою.

Підприємство повинно бути переконане у наявності міцних та надійних відносин зі своїми бізнес-партнерами для забезпечення максимальної безпеки в кіберпросторі. Основою діяльності кожної компанії має бути розробка процедур та впровадження відповідної моделі управління ризиками. Також важливим елементом є контроль і постійне вдосконалення стандартів кібербезпеки.

Важливим є розробка моделі, що дозволить компанії проаналізувати свою поточну ситуацію, визначити мережу взаємовідносин та дії, необхідні для виконання. Цей аналіз слід проводити систематично.

Першим кроком необхідно мати належне розуміння того, що повинно підлягати захисту, та чому може бути важко побудувати правильні відносини з бізнес-партнерами.

Важливим є розуміння причин атак на партнерську мережу; прогнозування ймовірних зловмисників та їх мотивації; дослідження існуючих вразливостей та загроз, ймовірних збитків та втрат від них; визначення ключових осіб організації, що впливатимуть на створення нової стратегії безпеки.

Щоб внести зміни та зробити їх тривалими, організація повинна розглянути: співробітників, що повинні підлягати змінам; відповідальних за безпеку ланцюжка постачання; перелік осіб, що мають право брати участь в обговоренні при створенні стратегії безпеки та володіють інформацією про зміни та прогрес в роботі. Завдяки цьому організація буде мати можливість ефективно спілкуватися зі співробітниками на відповідних посадах і визначати процес змін, враховуючи відповідні ролі, відповідальність та нагляд за процесом.

Важливо також визначити порядок оцінки ризиків ланцюжка постачання. Без належної оцінки ризику та наслідків їх застосування зловмисниками, неможливо буде запровадити відповідні зміни.

Другий крок полягає у розробці підходу до перевірки кібербезпеки ланцюжка постачання.

Необхідно розставити пріоритети для організації, визначивши активи, які потребують найбільшого захисту. Після визначення критичних елементів, які необхідно захистити, буде легко розробити процедури безпеки та налаштувати критерії для постачальників.

Визначення ключових елементів процедур перевірки включають: профілі безпеки кожного постачальника та визначення рівня їх безпеки на основі анкети; мінімальні вимоги безпеки для кожного профілю, які повинні бути виконані; план управління політикою безпеки, що включає перевірку постачальників на відповідність вимогам безпеки та визначення можливості подальшої співпраці при невідповідності будь-якої з вимог; нові положення про кібербезпеку, що будуть включені в контракти з постачальниками програмного забезпечення.

Третім кроком є застосування нового підходу до відносин з постачальниками.

Цей крок включає навчання команди. Необхідно переконатися, що ті, хто братиме участь в оцінці постачальників, пройшли відповідне навчання з кібербезпеки.

Також важливим є включення регулярного контролю кібербезпеки протягом усього терміну дії контрактів з постачальниками. Кібербезпека повинна враховуватися при прийнятті рішень про аутсорсинг, виборі постачальників, підписанні контрактів до їх розірвання або припинення. Тільки це забезпечить компанії повний контроль над бізнес-процесами. Варто також розглянути, які практики можна запровадити у випадку придбання продукції.

Необхідно контролювати рівень безпеки постачальників та доповідати керівництву про прогрес.

Четвертим кроком є інтегрування підходу до існуючих постачальників і контрактів. Він полягає у визначенні існуючих постачальників; оцінці поточних контрактів щодо ризиків, які вони несуть; підтримці своїх партнерів у процесі адаптації до сучасних вимог; перевірці договорів та положень до них; контролі рівня безпеки постачальників; доповіді керівництву про прогрес.

П’ятий крок полягає у постійній перевірці результатів.

Коли новий підхід до захисту мережі буде створено, варто подбати про безперервність. Необхідно постійно перевіряти дію внесених змін та їх якість; підтримувати обізнаність про нові загрози та відповідно оновлювати політику своєї організації, щоб забезпечити безпеку ланцюжка постачання; працювати з постачальниками для досягнення найкращих результатів.

Розглянемо мотивацію кіберзлочинців щодо атак на ланцюжки постачання.

Згідно з існуючими дослідженнями, більшість інцидентів були мотивовані бажанням отримати дані та інтелектуальну власність компанії. Крім того, відповідальні за половину цих інцидентів організовані групи кіберзлочинців. Вони мають відповідне обладнання, а також засоби та кваліфікацію для здійснення ефективного нападу. Це є дуже небезпечним для інформаційної системи цільового підприємства.

Варто розуміти, що масштаби даної загрози будуть тільки зростати. Досвідчені зловмисники використовують різноманітні методи. Одним із найпопулярніших з них є використання malware (malicious software) – шкідливого програмного забезпечення, яке може заразити практично будь-який пристрій, підключений до мережі Інтернет. Його метою може бути пошкодження функціональності системи, відкриття backdoor для подальших атак, блокування пристроїв або крадіжки даних.

Інші методи, які використовують кіберзлочинці, включають: виявлення вразливостей програмного забезпечення, методи соціальної інженерії, атаки методом «грубої сили» або використання проблем конфігурації програмного забезпечення. Тому й слід приділяти особливу увагу діяльності, пов’язаній із захистом ланцюжків постачання, а також постійному управлінню ризиками з боку організації.

4.1 Дослідження методів забезпечення безпеки на кожному етапі розробки програмного забезпечення

Групи безпеки, розробники та користувачі програмного забезпечення стикаються з трьома основними перешкодами, що заважають їм належним чином захистити себе або свою організацію від атак на ланцюжки постачання програмного забезпечення.

Першою перешкодою є відсутність збору й аналізу даних, які б допомогли визначити й оцінити ризики, пов’язані з цими атаками. Особливо це стосується використання програмного забезпечення з відкритим кодом.

Дана перешкода існує, незважаючи на публічний характер розробки програмного забезпечення з відкритим кодом і повсюдність залежностей з відкритим кодом у сучасному програмному забезпеченні. Тобто, значна частина даних, необхідних для визначення потенційних і фактичних ризиків, пов’язаних із компонентами програмного забезпечення, розміщена на загальнодоступних платформах розробки, таких як GitHub, і тому доступна будь-якій зацікавленій стороні. Через те, що значна частина цієї інформації не аналізується, зловмисники мають можливість ховатися на виду.

Щоб ідентифікувати атаки, захисникам потрібно відбирати та аналізувати дані, пов’язані з розробкою програмного забезпечення. Для початку потрібно виконати опис матеріалів програмного забезпечення, що містить всі залежності програми, забезпечує прозорість і дозволяє досліджувати прямі та непрямі залежності.

Іншими багатими джерелами даних є сховища коду з відкритим вихідним кодом і реєстри пакетів, які містять інформацію про плинність розробників, фіксацію коду та випуски версій, що зберігаються в цих сховищах.

Захисники також можуть виявляти невідповідності між незалежними джерелами даних, перевіряючи, наприклад, зв’язки між вихідним кодом, що зберігається в репозиторії, і випущеною бібліотекою або виконуваним файлом, що зберігається в реєстрі пакетів.

Протидія цьому також, ймовірно, вимагатиме розуміння осіб та організацій, які прямо чи опосередковано сприяють розробці та розповсюдженню програмного забезпечення, особливо осіб або організацій, які можуть публікувати зміни. Важливо, що захисникам знадобиться ця інформація для всіх залежностей поширюваного програмного додатку, незалежно від того, чи є ці залежності

частиною процесу збірки, випуску, чи включені під час виконання. Як завжди, джерелом ризику є залежність, а ризик, на відміну від вигоди, є транзитивним.

Другою перешкодою є те, що існуючі продукти безпеки додатків не можуть визначити відмінні характеристики атак на ланцюжки постачання програмного забезпечення. Крім того, існує обмежене впровадження того, які інструменти та процеси існують, щоб запобігти випадкам атак на ланцюжки постачання в рамках випущеного програмного забезпечення.

Описані проблеми змушують розробників програмного забезпечення та користувачів довіряти постачальникам та їхнім продуктам, але не перевіряти їх, унеможливлюючи судження про якість ланцюжка постачання програмного забезпечення продукту та змушуючи приймати невідомі ризики в критичному програмному забезпеченні.

Дані недоліки повинні об’єднати тих, хто хоче розробити та створити новий клас продуктів безпеки додатків, інструментів, призначених для виявлення випадків атак на ланцюжки постачання програмного забезпечення.

Існуючі інструменти безпеки програми призначені для виявлення дефектів у вихідному коді або виконуваних файлах та визначення умов, за яких ці дефекти можна використовувати. Однак ці інструменти не виявлять добре написаного компрометування в ланцюжку постачання програмного забезпечення. Такі атаки виникають через те, що цільове призначення програмного забезпечення має небажану функціональність.

Цей новий вид продуктів безпеки додатків потребуватиме контексту та розуміння очікуваного варіанту використання програми, концепцій, яких бракує поточним продуктам безпеки додатків, що є нелегкою задачею.

Третьою перешкодою, що заважає належним чином захистити себе або свою організацію від атак на ланцюжки постачання програмного забезпечення є те, що зменшення площі атаки на ланцюжок постачання програмного забезпечення також вимагає впровадження існуючих технологій і процесів, які надають інформацію, необхідну для перевірки походження та вмісту вихідного коду та двійкових файлів, усуваючи або пом’якшуючи багато ризиків компрометації. Одним із практичних кроків є використання цифрових підписів і сертифікатів для перевірки цілісності файлів. Використання відтворюваних збірок і публікація відповідних метаданих у незмінній розподіленій книзі також може дозволити споживачам самостійно перевіряти цілісність програмного компонента. Також слід розгортати в середовищі розробки, публікації та експлуатації найкращі у своєму роді

інструменти для захисту мережі та кінцевих точок, щоб обмежити можливості для компрометування перед фіксацією.

Зрештою, захист ланцюжків постачання програмного забезпечення будь-якого продукту вимагає постійної оцінки компонентів, постачальників і робочого середовища на додаток до аранжування та аналізу відповідних даних. Щоб ці процеси були успішними, потрібні значні інвестиції в автоматизацію та співпрацю між усіма учасниками ланцюжка постачання програмного забезпечення. Це необхідно для того, щоб спільне використання програмних залежностей стало перевагою, а не проблемою, якою воно є сьогодні.

Розглянемо, на які вразливості системи безпеки слід звернути увагу розробникам під час розробки програмного забезпечення. Опишемо три з них, що зустрічаються найчастіше. Вони базуються на проекті OWASP Top 10, який є галузевим еталоном найбільш поширених вразливостей.

Першою найпоширенішою вразливістю системи безпеки є ін’єкція.

Міжсайтовий сценарій (XSS) і ін’єкції SQL – це два існуючі типи вразливостей ін’єкцій, що найчастіше використовуються зловмисниками.

Уразливості XSS виникають, коли зловмисник може впровадити шкідливий код на веб-сайт, який в подальшому виконується користувачами, що нічого не підозрюють.

XSS є серйозною уразливістю і може призвести до того, що зловмисник зможе видавати себе за інших користувачів, шляхом отримання несанкціонованого доступу до їх облікових записів, спостерігати за поведінкою користувачів, завантажувати зовнішній вміст, викрадати конфіденційні дані тощо.

Щоб захиститися від атак, що використовують уразливість XSS, необхідно переконатися, що виконується блокування будь-якого виконуваного коду від будь-якого введення користувачем.

Атака SQL-ін’єкції полягає у вставці SQL-запиту через дані, що були введені в систему. В результаті успішного проведення даної атаки, з’являється можливість зчитування конфіденційних даних, що зберігаються в базі даних, зміни даних в базі даних, виконання адміністративних операцій з нею (таких як завершення роботи системи управління базами даних (СУБД)), відновлення вмісту файлу, наявного в СУБД і в деяких випадках видачі команди операційній системі.

Захист від атак SQL-ін’єкції передбачає встановлення заходів безпеки, подібних до тих, що застосовуються для захисту від атаки XSS.

Завдяки використанню наступних методів можна запобігти успішній реалізації SQL-ін’єкцій.

1) Використовувати підготовлені вирази (з параметризованими даними) при генеруванні запитів до бази даних.
2) Використовувати належним чином побудовані збережені процедури (цей підхід має той самий ефект, що й використання параметризованих запитів, якщо збережені процедури реалізовані безпечно.
3) Виконувати перевірку вхідних даних за списком дозволів (Allow-list Input Validation).
4) Використовувати принципи мінімізації привілеїв.

Другою найпоширенішою вразливістю системи безпеки є вразливі та застарілі компоненти.

Відомо, що сучасне програмне забезпечення створено на основі кодової бази сторонніх розробників, яка часто становить понад 80-90 % загального коду. У багатьох випадках зловмисники мають можливість швидко з’ясувати, яку версію компонента використовує дана програмна система. Крім того, у них під рукою є бібліотека легкодоступних вразливостей і експлойтів для цих компонентів. Отже, необхідні лише базові технічні навички, щоб успішно використовувати ці системи.

Захист від цих ризиків простий: необхідно завжди оновлювати компоненти програмного забезпечення до останніх версій. На жаль, на практиці все не так просто, оскільки дані компоненти можуть мати невиправлені залежності, проблеми із зворотною сумісністю або бути залишеними, і це лише деякі з них. Очевидно, щоб керувати сторонніми компонентами, спочатку потрібно виміряти їх, створивши специфікацію програмного забезпечення.

Третьою вразливістю є порушений контроль доступу. В результаті зловмисник має можливість обходу або використання слабких місць в механізмах автентифікації або авторизації програми. Це може призвести до несанкціонованого доступу до конфіденційних даних або систем.

Спосіб захисту від вразливостей автентифікації полягає у використанні багатофакторної автентифікації.

Щоб захиститися від уразливості авторизації, необхідно перевірити дозволи для кожного запиту та встановити правила повноважень, які за замовчуванням забороняють і приймають лише винятки.

Розглянемо, як забезпечити дотримання вимог безпеки протягом життєвого циклу розробки програмного забезпечення.

Вкрай важливо, щоб на етапі планування та аналізу життєвого циклу розробки захищеного програмного забезпечення було визначено вимоги до безпеки, які є специфічними для програмного забезпечення, що розробляється. Для цього може знадобитися визначити нормативні вимоги (наприклад, вимоги відповідності General Data Protection Regulation (GDPR)), найкращі практики безпеки та організаційну політику, яких потрібно дотримуватися.

Стандарт перевірки безпеки додатків OWASP (ASVS) є галузевим стандартом для вимог безпеки, і це чудова відправна точка для будь-якого програмного продукту.

Необхідно включити безпеку на етапі проектування.

Безпека також повинна бути ключовим аспектом дизайну. Це може включати розробку моделей загроз, виявлення потенційних вразливостей і розробку засобів контролю безпеки для боротьби з цими вразливими місцями. Моделювання загроз передбачає ідентифікацію та оцінку потенційних загроз програмної системи, що допомагає розробити стратегії пом’якшення цих загроз.

Розробники також повинні використовувати безпечні методи кодування.

По-перше, необхідно реалізовувати процес перевірки вхідних даних програми та заборони тих, які не відповідають певним критеріям, дозволяючи тільки вхідні дані, які їм відповідають. Це значно ускладнить для зловмисника введення вхідних даних, що призначені для нанесення шкоди системі.

По-друге, провести кодування виводу, що представляє собою переклад спеціальних символів у іншу, але еквівалентну форму, яка більше не є небезпечною для цільового інтерпретатора. Це служить формою захисту від міжсайтового сценарію.

По-трете, виконати хешування пароля, що значно ускладнить відновлення пароля у разі порушення безпеки.

Практики безпечного кодування є частиною методології DevOps.

Необхідно проводити тестування безпеки протягом усього процесу розробки, щоб виявити та усунути вразливості, що згадувалися раніше.

Після випуску програмного забезпечення важливо стежити за проблемами безпеки, усуваючи виявлені вразливості. Це може включати впровадження виправлень безпеки, проведення додаткового тестування безпеки або внесення змін до архітектури чи дизайну програмного забезпечення.

Забезпечити дотримання вимог безпеки протягом життєвого циклу розробки програмного забезпечення також допоможе проведення періодичних перевірок

безпеки для того, щоб переконатися, що вимоги безпеки виконуються та програмне забезпечення є захищеним. Це можна зробити, шляхом проведення внутрішніх перевірок спеціальною групою безпеки або можна найняти сторонню перевірку безпеки.

Вимоги до безпеки та передовий досвід постійно вдосконалюються. Тому важливо безперервно контролювати та покращувати безпеку програмного забезпечення протягом усього його життєвого циклу, щоб переконатися, що воно залишатиметься безпечним протягом тривалого часу. Наприклад, це передбачає безперервне тестування програмного забезпечення на наявність вразливостей і виправлення їх за допомогою патчів безпеки.

Також важливо слідкувати за новими розробками в галузі безпеки програмного забезпечення, щоб забезпечити врахування найновішої інформації під час вдосконалення програмного забезпечення.

Розглянемо, яким чином тестування безпеки можна інтегрувати в процес розробки програмного забезпечення.

Усі фази життєвого циклу розробки безпечного програмного забезпечення повинні інтегрувати тестування безпеки. Існують різні взаємодоповнюючі підходи та техніки для цього.

Першим підходом є автоматизоване тестування безпеки pipeline. Це процес сканування програми на наявність вразливостей за допомогою автоматизованих інструментів. Такі інструменти можна інтегрувати в процес створення та розгортання для автоматичного тестування на загальні вразливості. Цей підхід може допомогти виявити проблеми безпеки на ранніх етапах процесу розробки, коли їх легше вирішити. Крім того, це може бути особливо корисним для виявлення повторюваних проблем.

Інструменти автоматичного тестування безпеки зазвичай легко виявляють такі типи вразливостей:

– ін’єкції: такі як XSS або ін’єкції SQL;
– переповнення буфера, що представляє собою запис за межі виділеної пам’яті та може призвести до пошкодження даних, збою програми або спричинити виконання шкідливого коду;
– слабкі алгоритми шифрування, що можуть призвести до розкриття конфіденційних даних, витоку ключа шифрування, несправної автентифікації, незахищених сеансів і атак спуфінгу (алгоритми MD5 і RC4 є прикладами слабких алгоритмів шифрування).

Тестування на проникнення – це метод тестування, що передбачає спробу порушити частину або весь захист системи. Таке тестування зосереджено на використанні тих самих інструментів і методів, що можуть використовуватися зловмисниками. Ці тести можуть проводитися спеціалізованими тестувальниками безпеки або розробниками, що були навчені методам тестування безпеки. Зрештою це допоможе виявити проблеми безпеки, що не були виявлені автоматизованими інструментами.

Безперервна інтеграція та доставка Pipeline також є підходом для інтегрування тестування безпеки в процес розробки програмного забезпечення. Практика DevOps безперервної інтеграції та доставки (CI/CD) Pipeline зосереджена на частій і надійній доставці програмного забезпечення та ділиться на дві частини.

Безперервна інтеграція – це процес, за допомогою якого розробники можуть самостійно працювати над власною «гілкою» кодування, щоб внести невеликі зміни в код. Потім ці зміни надсилаються в автоматизовану систему, яка створює та тестує зміни коду, перш ніж вони будуть додані до головного коду.

Після завершення процесу безперервної інтеграції зміни коду поміщаються в сховище коду. Потім команда операцій може розгорнути їх у робочому середовищі (де остання версія програмного забезпечення надсилається користувачам).

Наступним кроком проводиться тестування програмного забезпечення, а знайдені помилки виправляються за допомогою автоматизованих процесів. Після цього команда DevOps отримує сповіщення про останню збірку, яку вони можуть вручну надіслати на етап розгортання.

Необхідно також інтегрувати тестування безпеки в конвеєр CI/CD, щоб виявити та усунути проблеми безпеки перед випуском оновлень. Часте надсилання коду може допомогти швидко усунути вразливі місця в новому коді, забезпечуючи якість програмного забезпечення.

Програми Bug bounty передбачають надання компенсації особам за повідомлення про помилки, особливо ті, що стосуються питань безпеки. Це допомагає незалежним дослідникам безпеки виявляти вразливі місця в програмному забезпеченні та повідомляти про них. Крім того, традиційні методи безпеки можуть не виявити всі вразливості, тому такий підхід може допомогти виявити додаткові з них. На відміну від тесту на проникнення, програми Bug bounty зазвичай зосереджуються на фактичному використанні вразливостей.

Огляд коду та аудит можуть допомогти покращити безпеку програмних систем кількома способами, наприклад, це може допомогти виявити вразливі місця в безпеці.

До вразливостей, які нелегко виявити автоматизованими інструментами входять:

– порушений контроль доступу, який наразі є найпоширенішою вразливістю в списку Топ-10 OWASP;
– незахищений дизайн, що відноситься до широкої категорії слабких місць, виражених як «відсутній або неефективний дизайн контролю»;
– неправильна конфігурація безпеки, яка представляє купу проблем, пов’язаних із посиленням безпеки, непотрібними відкритими портами та службами, обліковими записами за замовчуванням тощо.

Важливим є забезпечення дотримання стандартів кодування. Для того, щоб переконатися, що код відповідає стандартам та безпечним методам кодування можна застосувати огляд і аудит коду.

Покращити безпеку програмних систем також допоможе поліпшення якості програмного забезпечення. Для загального покращення якості ПЗ можна використовувати перевірку коду та аудит, виявляючи та виправляючи дефекти на ранній стадії розробки. Це може зменшити ризик уразливості безпеки та підвищити надійність програмного забезпечення.

Навчання розробників також є одним з засобів покращення безпеки програмних систем.

Однією з найбільш занедбаних проблем є освіта. Кожен розробник повинен пройти щонайменше 50 годин навчання з безпеки на рік. Огляд коду та аудит, очевидно, можуть допомогти розробникам зрозуміти проблему та прогалини.

Нарешті, аналіз коду та аудит забезпечують зворотний зв’язок з розробниками програмного забезпечення щодо їх коду. Це може містити пропозиції щодо покращення та виявлення потенційних проблем безпеки.

Додатковою практикою, яку необхідно застосувати, є використання контейнерів. Контейнери – це пакети програми та її залежностей. Ізоляція цих контейнерів від хост-системи та інших контейнерів може ефективно зменшити ризик вразливості безпеки [1].

4.2 Слабке і шкідливе програмне забезпечення та безпека ланцюжка постачання програмного забезпечення

Хоча результати є неоднозначними, все більше зусиль докладається для створення безпеки в програмному забезпеченні.

Потреба в безпеці, очевидно, перевищує здатність і бажання інженерів програмного забезпечення розробляти безпечні архітектури програмного забезпечення, впроваджувати безпечні методи кодування, виконувати функціональне тестування безпеки та ретельно керувати встановленням програмних продуктів на різних платформах і в різних середовищах.

Програмне забезпечення COTS/GOTS (commercial off-the-shelf (COTS) та government off-the-shelf (GOTS)) часто містить численні вразливості (таке програмне забезпечення можна вважати «слабким програмним забезпеченням»), і воно іноді містить шкідливе програмне забезпечення.

Основна відмінність між слабким програмним забезпеченням і зловмисним програмним забезпеченням полягає в тому, що недоліки першого є здебільшого ненавмисними або випадковими, тоді як шкідливі характеристики другого є запланованими і навмисними та зазвичай потребують певної міри технічного досвіду для ефективного впровадження.

Тим не менш, з точки зору безпеки потенційні наслідки є небажаними та шкідливими, незалежно від того, як слабкий або поганий код взагалі потрапив у програму. Тому важливо дослідити, як і де такий шкідливий код або програми можуть бути введені протягом життєвого циклу постачання програмного забезпечення, та як можна уникнути таких слабких і зловмисних програм, стримати їх, усунути або пом’якшити ризики їх використання.

Необхідно враховувати конкретні загрози, пов’язані з програмним забезпеченням, які є набагато менш відчутними щодо їх впливу. До них відноситься несанкціонований і зловмисний доступ до інтелектуальної власності та конфіденційних даних, а також пошкодження або знищення програм або даних. Крім того, проблеми з піратством виникають через фальшиве програмне забезпечення.

Управління ризиками ланцюжка постачання зазвичай спрямоване на обмеження ризику збоїв, як правило, тих, які затримують доставку продукту виробнику або споживачу програмного забезпечення.

Однак управління ризиками ланцюжка постачання також використовується для зменшення або усунення підробленого та/або шкідливого програмного

забезпечення протягом життєвого циклу ланцюжка постачання, коли інсайдери та інші особи можуть мати доступ до програмного забезпеченнях [22].

Ризик постачання програмного забезпечення стосується як впливу несприятливих подій, так і ймовірності їх виникнення.

Серед потенційних наслідків можна виділити:

– перешкоджання задоволення покупцем попиту споживача;
– перешкоджання надання договірної технічної та операційної підтримки постачальником;
– порушення процесів управління ланцюжком постачання, зокрема, спричинення збоїв та затримок у дотриманні термінів, перенаправлення та крадіжки продуктів, несанкціоноване копіювання та розповсюдження тощо.

Розглянемо типи програмного забезпечення:

– програмне забезпечення COTS/GOTS;
– програмне забезпечення з відкритим кодом;
– індивідуальне програмне забезпечення (розроблене всередині, зовні або обидва);
– гібридне поєднання індивідуального програмного забезпечення з відкритим вихідним кодом і готового (OTS – Off-the-Shelf) програмного забезпечення;
– вбудоване програмне забезпечення – програмне забезпечення або вбудоване програмне забезпечення, що вбудоване у фізичні продукти;
– програмне забезпечення для керування ланцюжком постачання – спеціалізована категорія програмного забезпечення, яке відстежує процеси ланцюжка постачання і повідомляє про відхилення від очікуваної поведінки.

Стосовно доступу до вихідного коду клієнтами або іншими сторонами програмні продукти є закритими або відкритими.

Зазвичай вихідний код готового програмного забезпечення недоступний для клієнтів.

Користувачі як програмного забезпечення з відкритим вихідним кодом, так і індивідуального програмного забезпечення зазвичай мають доступ до вихідного коду, що дозволяє переглядати код і статичне тестування.

Загалом гібридне програмне забезпечення слід вважати настільки ж слабким, наскільки слабким є його найслабший компонент.

Дослідження показали, що програмне забезпечення з відкритим кодом може мати стільки ж серйозних проблем безпеки, скільки й програмне забезпечення COTS/GOTS.

Навіть незважаючи на те, що вбудоване програмне забезпечення може бути не головним завданням для виробників і розповсюджувачів фізичних продуктів, необхідно враховувати фактори ризику, пов’язані з програмним забезпеченням.

Програмне забезпечення, яке використовується для керування ланцюжками постачання (незалежно від того, чи це ланцюжки постачання для програмного забезпечення, фізичних об’єктів або комбінацій програмного та апаратного забезпечення), також необхідно враховувати, оскільки ефективне управління ланцюжками постачання може пом’якшити багато факторів ризику, які зазвичай виникають.

Програмне забезпечення для керування ланцюжком постачання також може бути вектором для кіберзловмисників.

Розглянемо характеристики ризику програмного забезпечення.

Програмне забезпечення відрізняється від промислових продуктів кількома важливими ознаками, а саме:

– програмне забезпечення можна вкрасти без необхідності видалення або іншим чином змінити оригінальну копію;
– фізичне транспортування програмного забезпечення не потрібне, оскільки копії можна завантажити в електронному вигляді;
– дійсні та законні версії програмного забезпечення можна змінювати та видавати за дійсні, навіть якщо їх було змінено.

Для фізичних продуктів зазвичай переважають ризики, пов’язані з виробництвом і розповсюдженням.

Коли справа доходить до програмного забезпечення, більше уваги потрібно приділяти раннім етапам процесу розробки продукту, таким як фази проектування та вимог, оскільки виробництво та розповсюдження є набагато меншими частинами загального ланцюжка постачання програмного забезпечення, ніж для фізичних продуктів [23].

Наступні атрибути ланцюжка постачання програмного забезпечення знаходяться під загрозою [24]:

– конфіденційність (інтелектуальна власність, особиста інформація та бізнес-дані);
– цілісність (процеси, продукти та дані);

– доступність (потоки, продукти та дані);
– автентичність (продукти та дані);
– надійність (процеси, продукти та люди).

Що стосується конфіденційності, необхідно не тільки враховувати потенційний ризик викрадення інтелектуальної власності та комерційних таємниць, але й усвідомлювати можливі наслідки компрометації особистих даних клієнтів і співробітників.

Коли мова заходить про цілісність, можна уявити, що самі процеси ланцюжка постачання використовуються злочинцями, а також виконується модифікація програмних продуктів і пов’язаних даних.

Потрібно також продемонструвати, що зусилля щодо пом’якшення наслідків були ефективними.

Наступні властивості, такі як прозорість, якість і підзвітність, дозволяють мати більшу впевненість у тому, що ризики були належним чином зменшені.

Учасники ланцюжка постачання піддаються різним загрозам у цьому ланцюжку.

Результат того, як загрози можуть бути розподілені по складових ланцюжка постачання наведено в таблиці 4.1.

Таблиця 4.1 – Віднесення загроз до складових ланцюжка постачання

Загрози	Складові ланцюжка постачання					
	Процеси ланцюжка постачання	Потоки даних ланцюжка постачання	Продукти	Потоки продукту	Дані керування	Люди
Внутрішня загроза	+	+	+	+	+	+
Диверсія	-	+	-	+	+	-
Ексфільтрація – зрив	+	+	-	-	+	-
Ексфільтрація – крадіжка	-	+	-	+	+	-
Знищення/ видалення	-	+	+	-	+	-
Зрив/затримка	+	+	-	+	+	-
Іноземна власність, вплив	+	+	+	+	+	+
Корупція	+	+	-	-	+	+
Крадіжка	+	+	+	-	+	-
Небажані фізичні елементи	-	-	+	+	-	+
Підробка, піратство	-	+	+	-	+	+
Порушення експортного контролю	+	-	-	+	-	+
Інфільтрація, «підривна діяльність»	+	+	-	-	-	

Продовження табл. 4.1

Загрози	Складові ланцюжка постачання					
	Процеси ланцюжка постачання	Потоки даних ланцюжка постачання	Продукти	Потоки продукту	Дані керування	Люди
Псевдоінсайдерська загроза	+	+	+	+	+	+
Саботаж	+	+	+	-	+	-
Соціальна інженерія	+	+	-	+	+	+
Фальсифікація	-	+	+	-	+	-

В таблиці 4.1 можна побачити результат віднесення загроз до складових ланцюжка постачання. В даній таблиці наведено такі складові ланцюжка постачання як: процеси ланцюжка постачання, потоки даних ланцюжка постачання, продукти, потоки продукту, дані керування та люди, а також загрози, що можуть відноситися до даних складових.

У звіті SEI (Software Engineering Institute) вказується на схожість і відмінності між постачальниками продуктів і підрядниками з розробки систем [25].

У даному звіті зазначається, що еквайери оцінюють програмне забезпечення після завершення розробки продукту, тоді як для систем, створених на замовлення, еквайери можуть «активно контролювати ризики підрядника та ланцюжка постачання продукту в процесі розробки».

У звіті SEI пропонується, щоб аналіз ризиків включав такі три компоненти: аналіз атак, тобто аналіз загроз і експлойтів, що призводить до успішних атак; можливість обмеження вразливих місць продукту постачальником; ідентифікацію «засобів атаки» та бізнес-ризиків покупцем.

Залежно від різних етапів ланцюжка постачання або життєвого циклу придбання змінюватимуться й принципи управління ризиком.

Управління ризиками безпеки протягом всього життєвого циклу необхідне для забезпечення захищеності ланцюжка постачання. Тому важливо розглянути,

яку діяльність з управління ризиками безпеки ланцюжка постачання необхідно виконувати за фазами SDLC.

Принципи управління ризиками безпеки ланцюжка постачання за фазами SDLC наведено в таблиці 4.2.

Таблиця 4.2 – Принципи управління ризиками безпеки ланцюжка постачання за фазами SDLC

Фаза SDLC	Діяльність з управління ризиками
Вимоги та дизайн	– провести оцінку ризику; – встановити вимоги безпеки; – розробити плани аудиту.
Виробництво (розробка)	– провести моніторинг процесів та потоків продукції; – виконати перевірку, тестування, верифікацію та підтвердження кінцевих продуктів.
Дистрибуція	– провести моніторинг процесів та потоків продукції.
Зберігання	– провести моніторинг процесів та потоків продукції; – перевірити, чи продукт не було видалено, змінено чи додано.
Розгортання	– провести моніторинг процесів та потоків продукції; – перевірити правильність та автентичність поставлених продуктів та систем; – надати користувачам інструкції, щоб переконатися, що продукти та системи не було підроблено чи іншим чином скомпрометовано.

Експлуатація	– відстежувати роботу на предмет незвичної поведінки та шкідливих подій; – проводити перевірку оперативної готовності на постійній основі;

Продовження табл. 4.2

Фаза SDLC	Діяльність з управління ризиками
	– виконати розробку та реалізацію плану реагування на інциденти безпеки.
Технічне обслуговування та підтримка	– відстежувати постачальників продуктів та компонентів на наявність будь-яких несприятливих звітів, що стосуються життєздатності компаній-постачальників або будь-яких проблем із безпекою продуктів; – провести розробку плану на випадок можливих збоїв у постачанні, наприклад, частин і патчів та підтримки.
Утилізація	– контролювати утилізацію інтелектуальної власності та конфіденційних даних, таких як особиста інформація, а також знищення носіїв, що містять таку інформацію.

В таблиці 4.2 можна побачити принципи управління ризиками безпеки ланцюжка постачання за фазами SDLC, такими як: вимоги та дизайн, розробка, дистрибуція, зберігання, розгортання, експлуатація, технічне обслуговування та підтримка, а також утилізація.

Наведемо фактори, що впливають на ризик ланцюжка постачання протягом життєвого циклу розробки.

Ризики ланцюжка постачання за фазами SDLC представлені на рисунку 4.1.

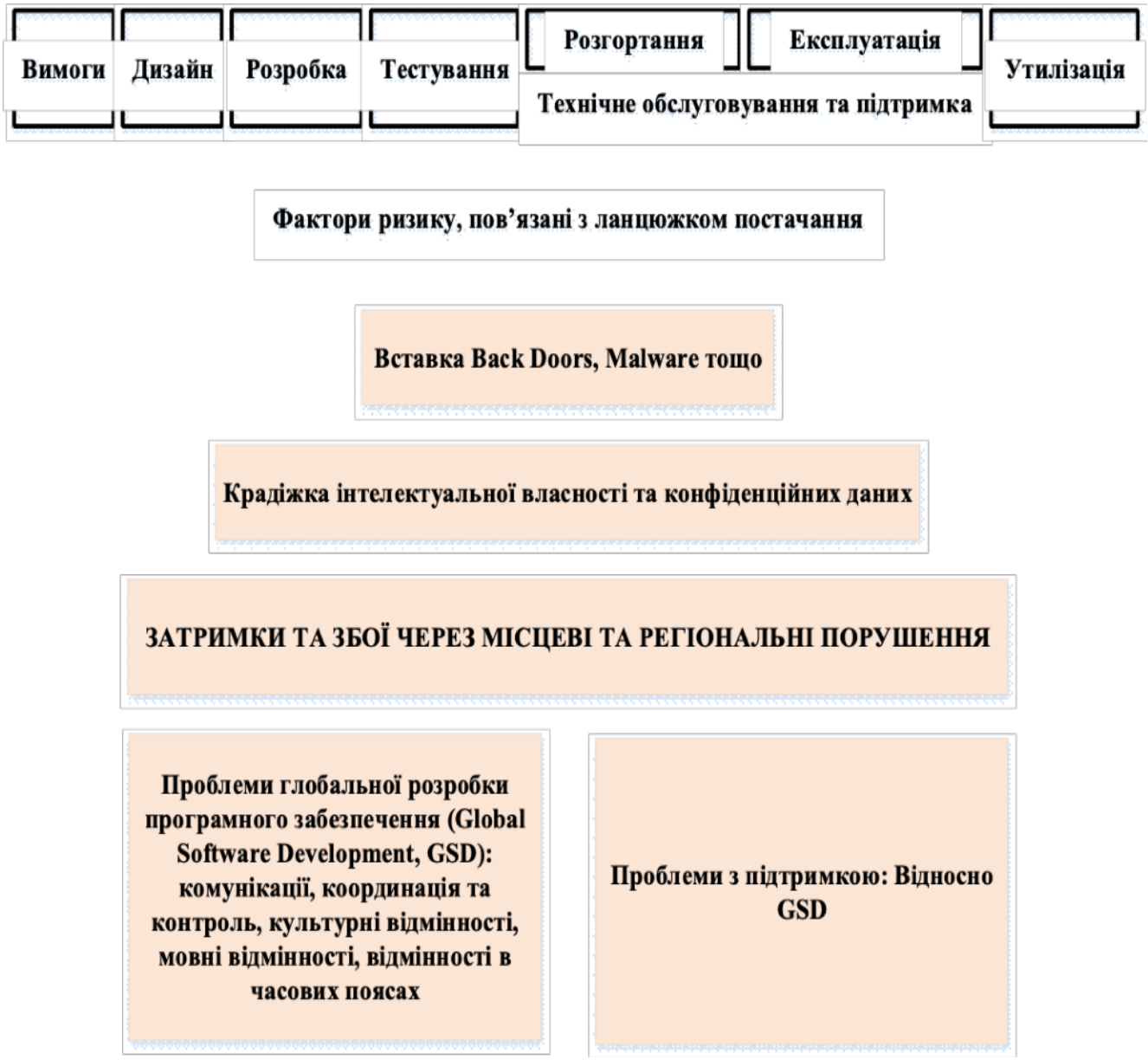

Рисунок 4.1 – Ризики ланцюжка постачання за фазами SDLC

На рисунку 4.1 можна побачити фактори, що впливають на ризик ланцюжка постачання протягом життєвого циклу розробки, що включає такі етапи як: вимоги, дизайн, розробка, тестування, розгортання, експлуатація, технічне обслуговування та підтримка, а також утилізація.

На рисунку 4.1 видно, що існують події, такі як природні та антропогенні катастрофи, які можуть вплинути на всі ланцюжки постачання, включаючи ланцюжки постачання програмного забезпечення.

Існує також низка компрометацій, таких як вставка зловмисного програмного забезпечення, які є унікальними для програмного забезпечення.

Інші інциденти, такі як крадіжка інтелектуальної власності та особистих даних, є поширеними для багатьох продуктів, але полегшуються для програмного забезпечення завдяки можливості копіювати програмне забезпечення та дані без зміни оригіналу або необхідності бути на місці для копіювання.

Оскільки глобальні ланцюжки постачання складаються з багатьох складових і компонентів, дуже важко отримати повне уявлення про всі їх складності та нюанси.

Можливо, єдиним ефективним засобом для того, щоб отримати повне уявлення про ланцюжки постачання є побудова комп'ютерних моделей процесів, потоків і засобів керування, а також використання їх у вправах для кращого розуміння взаємодії компонентів і впливу різних атак і подій.

Основною відмінністю факторів ризику, пов'язаних із проектуванням і розробкою програмного забезпечення, є розташування цих зусиль і культура тих, хто виконує роботу.

Розташування можна визначити з точки зору того, чи здійснюється проектування та розробка внутрішніми силами чи сторонніми виконавцями, а також за географічним розташуванням.

Рівень ризику сильно залежить від таких факторів, як: лояльність і мотивація співробітників і підрядників; правові, соціальні та економічні відмінності між країнами та етнічними групами тощо.

Вирішення проблем, пов'язаних із глобальною розробкою програмного забезпечення та ризиками, пов'язаними з ланцюжками постачання програмного забезпечення є елементами багатьох досліджень.

Моделювання необхідне для оптимізації різних характеристик розосереджених зусиль з розробки програмного забезпечення через складність та динамічність таких механізмів.

Дослідники розробили моделі впливу культурних, комунікаційних та інших факторів на розподілену розробку всіх типів програмного забезпечення. Наприклад, сектор фінансових послуг США працював над проблемами ланцюжка постачання та опитував представників галузі щодо різних аспектів зменшення ризиків ланцюжка постачання. Проте, як видається, немає особливого способу моделювання впливу несприятливих природних і антропогенних явищ на ланцюжки постачання програмного забезпечення. Незважаючи на те, що потреба в таких моделях очевидна, зусилля для розробки таких моделей є значними.

Розглянемо питання розробки та розповсюдження програмного забезпечення.

Щоб приймати рішення щодо будь-якого конкретного ланцюжка постачання, необхідно розуміти кожну фазу, а також принципи взаємодії між цими фазами.

Принципи дії фаз нагляду, виробництва та гарантії в SDLC представлені на рисунку 4.2.

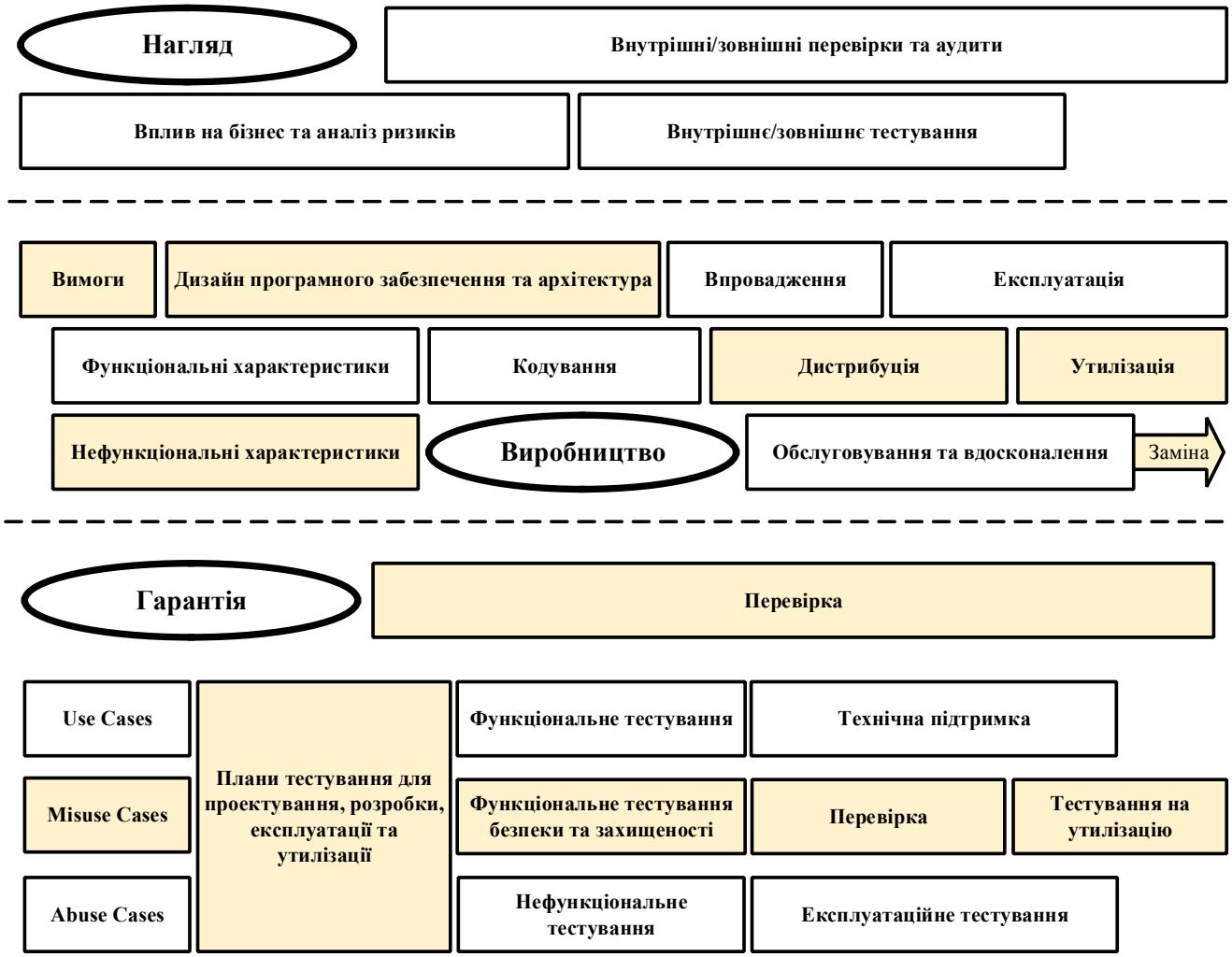

Рисунок 4.2 – Фази нагляду, виробництва та гарантії в SDLC

На рисунку 4.2 показано життєві цикли для трьох основних аспектів розробки, тестування та розгортання програмного забезпечення, таких як виробництво, нагляд і гарантія.

Виробництво – це практична основа розробки та розповсюдження програмного забезпечення.

Нагляд полягає в незалежному спостережені за процесами та продуктами життєвого циклу виробництва.

Гарантія включає окремі оцінки якості, цілісності та надійності програмного забезпечення, що виготовляється.

Виділеними елементами на рисунку 4.2 представляються функції, яким у SDLC часто приділяється недостатня увага.

Серед важливих сфер є тестування функціональної безпеки, тобто тестування, яке виконується для того, щоб переконатися, що програмне забезпечення не виконує непотрібних дій, а також дії, пов’язані з утилізацією програмного забезпечення та будь-якої конфіденційної інформації, яку воно може містити.

У той час як етапи верифікації та перевірки є звичайними компонентами життєвих циклів розробки, яких дотримується державні установи, вони часто не повністю розроблені в приватному секторі.

Ризики, пов’язані з ланцюжком постачання програмного забезпечення, значною мірою залежать від природи та походження програмного забезпечення.

Розглянемо рівні ризику, які можна очікувати щодо програмного забезпечення з різних джерел і незалежно від того, чи доступна технічна підтримка.

Рівні ризику ланцюжка постачання програмного забезпечення наведено в таблиці 4.3.

Таблиця 4.3 – Рівні ризику ланцюжка постачання програмного забезпечення

Ризики	Джерела			
	Програмне забезпечення COTS/GOTS	Програмне забезпечення з відкритим вихідним кодом	Індивідуальне програмне забезпечення	Програмне забезпечення, що не підтримується
Ризик того, що зловмисне програмне забезпечення було запроваджено на етапі розробки	Середній	Низький	Середній	Високий

Продовження табл. 4.3

Ризики	Джерела			
	Програмне забезпечення COTS/GOTS	Програмне забезпечення з відкритим вихідним кодом	Індивідуальне програмне забезпечення	Програмне забезпечення, що не підтримується
Ризик запровадження зловмисного програмного забезпечення під час експлуатації	Від середнього до високого	Від низького до середнього	Низький	Високий
Ризик того, що неправильна утилізація призведе до компрометування	Від середнього до високого	Від середнього до високого	Низький	Від низького до середнього

В таблиці 4.3 можна побачити рівні ризику ланцюжка постачання програмного забезпечення для таких джерел як: програмне забезпечення COTS/ GOTS, ПЗ з відкритим вихідним кодом, індивідуальне ПЗ та ПЗ, що не підтримується.

Початкові проблеми полягають у виявленні та розумінні загроз і вразливостей ланцюжка постачання програмного забезпечення та захисту від них.

Ризик необхідно розглядати на кожному етапі розробки програмного забезпечення та життєвих циклів ланцюжка постачання програмного забезпечення, а також необхідно аналізувати заходи, запропоновані для пом’якшення факторів ризику.

Єдиний спосіб повністю зрозуміти складні ланцюжки постачання – це розробити моделі комп’ютерного моделювання, які представляють ці ланцюжки постачання на рівні транзакцій, тобто з точки зору процесу та потоку продукту.

Багато чого вже було зроблено дослідниками в різних галузях для кращого розуміння ланцюжків постачання програмного забезпечення та притаманного їм

ризику. Проте ще багато чого потрібно зробити в державному та приватному секторах, щоб досягти прийнятного рівня розуміння пов’язаних ризиків та їх пом’якшення.

4.3 Оцінка захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення

Узагальнимо основні вразливості безпеки ланцюжка постачання. Результат наведено в таблиці 4.4.

Таблиця 4.4 – Основні вразливості безпеки ланцюжка постачання

Вразливості	Опис вразливості
Відсутність видимості	Великою проблемою є моніторинг, оскільки різні люди керують різними частинами ланцюжка, а команди та інструменти відрізняються. Для забезпечення безпеки необхідно зібрати всі дані з кожного кроку та систематизувати їх у єдиному місці.
Схильні до помилок ручні процеси	CI/CD насправді працює лише напівавтоматично з великою кількістю ручного втручання на кожному кроці, що сприяє виникненню уразливості, причиною якої є людська халатність або зловмисники усередині організації. При повній автоматизації процесів ланцюжка постачання, ризик полягає у встановленні засобів контролю та перевірок безпеки, відсутність яких може зробити автоматизацію шкідливою.
Складність ланцюжка постачання	Потрібні інтеграції були створені завдяки безперебійному конвеєру CI/CD за допомогою різних найкращих у своєму класі інструментів сторонніх розробників. Вони зобов’язані ввімкнути спеціальні робочі процеси за участю сторонніх постачальників і партнерів. Ці інтеграції є живильним середовищем для вразливостей і легко підбираються зловмисниками.

Продовження табл. 4.4

Вразливості	Опис вразливості
Неправильне поводження з секретами	Конфіденційна інформація має багато форм та не може бути жорстко закодована в програмі або зберігатися в незашифрованих файлах. Вони потребують спеціальної обробки створеними інструментами керування секретами. В протилежному випадку можуть виникнути порушення безпеки.
Відсутність практик безпеки API	API є складовою, що об'єднує хмарні системи, які є шлюзом сторонніх систем для доступу до послуг організації. Якщо вони скомпрометовані, легко отримати доступ до більш глибоких частин системи.
Уразливості у відкритому вихідному коді	Log4j є найновішою та найвідомішою з відкритих вразливостей. Розробникам не можна нехтували відкритим кодом. Відповідальність за забезпечення безпеки лежить на компанії, яка використовує ці інструменти з відкритим кодом.
Вразливе або невиправлене програмне забезпечення	Системи IoT і навіть багато застарілих систем програмного забезпечення не обслуговуються активно, а їх мікропрограмне або програмне забезпечення застаріло. Продовжувати перевіряти наявність застарілого програмного забезпечення та видаляти його з потрібної системи є складним, але необхідним процесом для забезпечення «герметичної» системи від початку до кінця.
Вразливі та застарілі компоненти	Через кодову базу сторонніх розробників, що використовується при розробці програмного забезпечення, у зловмисників часто є можливість швидко з'ясувати, яку версію компонента використовує дана програмна система. Також у них під рукою є бібліотека легкодоступних вразливостей і експлойтів для цих компонентів.

Продовження табл. 4.4

Вразливості	Опис вразливості
	Захист від цих ризиків полягає в регулярному оновлені компонентів програмного забезпечення до останніх версій. Щоб керувати сторонніми компонентами, спочатку потрібно виміряти їх, створивши специфікацію програмного забезпечення.
Неправильні налаштування інструмента CI/CD	Конфігурація, що передбачає кодифікацію конфігурації таких аспектів, як інфраструктура та політики, які керують програмними процесами зберігається у вигляді файлів YAML. Часто контроль над цими конфігураційними файлами не захищений належним чином, залишаючи вектори атак і вразливості відкритими для зловмисників. У чужих руках ці конфігураційні файли можуть бути зловживані.
Загрози вихідного коду (Незахищений код)	Причина – людська халатність та відсутність безпечних методів кодування. Незахищені робочі станції розробки можуть впроваджувати незахищений або шкідливий код. Пом’якшити загрози незахищеного коду допоможе реалізація політики на робочих станціях розробників, регулярне сканування коду і API під час і після розробки, а також запровадження методів безпечного кодування. Також необхідно переконатися, що розробники не мають можливості додавати незахищений або зловмисний код у сховища вихідних кодів, шляхом включення перевірки змін вихідного коду.
Загрози вихідного коду (Скомпрометовані системи контролю вихідного коду)	Код і репозиторії вихідних кодів можуть залишатися в безпеці, лише якщо конкретна система контролю вихідного коду не буде скомпрометована. Забезпечити безпеку можна шляхом обмеження доступу до цієї системи та інших систем у конвеєрі збірки.

Продовження табл. 4.4

Вразливості	Опис вразливості
	Можна використовувати MFA, щоб захистити доступ до системи та регулярно оцінювати інтеграцію джерела, включаючи конфігурації та сценарії.
Порушений контроль доступу	В результаті зловмисник має можливість обходу або використання слабких місць в механізмах автентифікації або авторизації програми, що може призвести до несанкціонованого доступу до конфіденційних даних або систем. Спосіб захисту від вразливостей автентифікації полягає у використанні багатофакторної автентифікації. Щоб захиститися від уразливості авторизації, необхідно перевірити дозволи для кожного запиту та встановити правила повноважень, які за замовчуванням забороняють і приймають лише винятки.

В таблиці 4.4 можна побачити основні вразливості безпеки ланцюжка постачання, а саме: відсутність видимості, схильні до помилок ручні процеси, складність ланцюжка постачання, неправильне поводження з секретами, відсутність практик безпеки API, уразливості у відкритому вихідному коді, вразливе або невиправлене програмне забезпечення, вразливі та застарілі компоненти, неправильні налаштування інструмента CI/CD, загрози вихідного коду (незахищений код та скомпрометовані системи контролю вихідного коду) та порушений контроль доступу.

Узагальнимо основні види атак на ланцюжки постачання. Результат наведено в таблиці 4.5.

Таблиця 4.5 – Основні види атак на ланцюжки постачання

Вид атаки	Опис атаки
Порушення CI/CD Pipeline	Зловмисники шукають способи проникнути в конвеєр CI/CD, який використовується організаціями для доставки програмного забезпечення. Оскільки конвеєр CI/CD є «центральною нервовою системою» всього процесу розробки програмного забезпечення, будь-які зміни, внесені тут, негативно вплинуть на виробничі програми, а також програми клієнтів.
Скомпрометовані інструменти для створення програмного забезпечення	Важливо захистити дані інструменти (від програмного забезпечення з відкритим кодом до комерційних інструментів, що зокрема створюють збірки, перевіряють якість і розгортають код у виробництві) та переконатися, що вони не стануть вектором впровадження шкідливого коду в конвеєр для загрозливого суб'єкта.
Атака плутанини залежностей	Останнім часом зловмисники знайшли спосіб обманом змусити розробників завантажувати шкідливі пакети, націлюючись на орфографічні помилки в пакетах, які найчастіше завантажуються. Оскільки розробники здебільшого вводять назви пакетів в інтерфейсі командного рядка – це призводить до помилок.
Міжсайтовий сценарій (XSS)	Може призвести до того, що зловмисник зможе видавати себе за інших користувачів, шляхом отримання несанкціонованого доступу до їх облікових записів, спостерігати за поведінкою користувачів, завантажувати зовнішній вміст, викрадати конфіденційні дані тощо. Для захисту необхідно переконатися, що виконується блокування будь-якого виконуваного коду від будь-якого введення користувачем.

Продовження табл. 4.5

Вид атаки	Опис атаки
Атака SQL-ін’єкції	Атака SQL-ін’єкції полягає у вставці SQL-запиту через дані, що були введені в систему. В результаті успішного проведення даної атаки, з’являється можливість зчитування конфіденційних даних, що зберігаються в БД, зміни даних БД, виконання адміністративних операцій з нею, відновлення вмісту файлу, наявного в СУБД і в деяких випадках видачі команди операційній системі. Щоб захиститися від цього, необхідно переконатися, що виконується блокування будь-якого виконуваного коду від будь-якого введення користувачем. Запобігти успішній реалізації SQL-ін’єкцій можна шляхом: використання підготовлених виразів (з параметризованими даними) при генеруванні запитів до бази даних; використання належним чином побудованих збережених процедур; виконання перевірки вхідних даних за списком дозволів; використання принципів мінімізації привілеїв.
Вставка backdoor, malware тощо	Malware – це шкідливе програмне забезпечення, яке може заразити практично будь-який пристрій, підключений до мережі Інтернет. Його метою може бути пошкодження функціональності системи, відкриття backdoor для подальших атак, блокування пристроїв або крадіжки даних.
Інші атаки	Інші методи, які використовують кіберзлочинці, включають: виявлення вразливостей програмного забезпечення, методи соціальної інженерії, атаки «грубою силою» або використання проблем конфігурації програмного забезпечення.

В таблиці 4.5 можна побачити основні види атак на ланцюжки постачання, а саме: порушення CI/CD Pipeline, скомпрометовані інструменти для створення

програмного забезпечення, атака плутанини залежностей, міжсайтовий сценарій (XSS), атака SQL-ін’єкції, вставка backdoor, malware, а також інші методи, що використовуються кіберзлочинцями – виявлення вразливостей програмного забезпечення, методи соціальної інженерії, атаки «грубою силою» або використання проблем конфігурації програмного забезпечення.

Результати систематизації методів захисту від загроз, що використовуються зловмисниками для атак на ланцюжки постачання програмного забезпечення наведено в таблиці 4.6.

Таблиця 4.6 – Систематизація методів захисту ланцюжків постачання програмного забезпечення

№ з/п	Метод безпеки	Ранг	Вид методу безпеки
1	Надання найменш привілейованого доступу до ресурсів у всьому ланцюжку постачання.	1	технічні
2	Увімкнення багатофакторної автентифікації, використання надійних паролів.	4	технічні
3	Регулярне навчання співробітників з безпеки.	14	організаційні
4	Отримання даних про постачальників та партнерів конкретної організації, починаючи з постачальників першого рівня.	16	організаційні
5	Проведення оцінки ризиків, щоб оцінити стан кібербезпеки кожного постачальника.	15	комплексні
6	Дотримання державної політики щодо вразливостей.	23	законодавчі
7	Регулярне сканування та виправлення вразливих систем.	11	технічні
8	Використання методів безпечного кодування, файлів блокувань та інших ініціатив, орієнтованих на безпеку:	3	технічні
	– перевірка контрольних сум;		
	– включення залежності від постачальника в систему керування джерелами;		

	– публікування та використання SBOM;		

Продовження табл. 4.6

№ з/п	Метод безпеки	Ранг	Вид методу безпеки
	– використання SLSA: можливість цифрового підпису артефактів програмного забезпечення для автентифікації походження та використання автоматизації для конкретних процесів і політик;		
	– сканування програмного забезпечення за допомогою автоматизованих інструментів тестування безпеки, таких як аналіз складу програмного забезпечення (SCA), статичне тестування безпеки додатків (SAST) і динамічне тестування безпеки додатків (DAST);		
	– реалізація процесу перевірки вхідних даних програми та заборони тих, які не відповідають певним критеріям, дозволяючи тільки вхідні дані, які їм відповідають;		
	– проведення кодування виводу;		
	– хешування пароля.		
9	Використання контрольних списків для контролю процесів.	5	технічні
10	Зменшення поверхні атаки, шляхом видалення старих та невикористовуваних інструментів та компонентів з ланцюжка постачання, зберігаючи кодову базу додатків невеликою та легкою, а також шляхом скорочення компонентів інфраструктури до тих, що використовуються на даний час; видалення непотрібних користувачів і обмеження їх права, виходячи з їх обов’язків та поставлених задач.	9	технічні

11	Регулярне сканування кожного кроку ланцюжка постачання.	17	технічні
12	Регулярне проведення тестування на проникнення.	18	технічні

Продовження табл. 4.6

№ з/п	Метод безпеки	Ранг	Вид методу безпеки
13	Проведення оновлення компонентів програмного забезпечення до останніх версій та створення специфікації програмного забезпечення. Проведення відбору та аналізу даних, що пов'язані з розробкою програмного забезпечення; виконання опису матеріалів програмного забезпечення, що містить всі залежності програми, забезпечує прозорість і дозволяє досліджувати прямі та непрямі залежності.	19	технічні
14	Використання графіків залежностей для зменшення ризиків безпеки.	24	технічні
15	Використання цифрових підписів і сертифікатів для перевірки цілісності файлів.	6	технічні
16	Проведення блокування будь-якого виконуваного коду від будь-якого введення користувачем.	2	технічні
17	Виконання перевірки дозволів для кожного запиту та встановлення правил повноважень, які за замовчуванням забороняють і приймають лише винятки.	7	технічні
18	Проведення автоматизованого тестування безпеки pipeline – сканування програми на наявність вразливостей за допомогою автоматизованих інструментів.	10	технічні
19	Виконання регулярного огляду коду та аудиту.	20	технічні
20	Проведення сегментації мережі.	13	технічні
21	Дотримання правил DevSecOps.	8	комплексні
22	Проведення захисту інфраструктури за допомогою безпечних шлюзів API, VPN, адміністративних панелей; управління доступом.	12	технічні

23	Реалізація фізичного захисту.	22	технічні

Продовження табл. 4.6

№ з/п	Метод безпеки	Ранг	Вид методу безпеки
24	Реалізація захисту носіїв інформації. Забезпечення підвищення безпеки всіх підключених пристроїв і конфіденційних даних.	21	технічні

В таблиці 4.6 можна побачити результат систематизації методів захисту ланцюжків постачання програмного забезпечення від атак.

Ранги та види методів захисту необхідні для того, щоб в подальшому використовувати ці дані для оцінки захищеності інформаційних систем організації від атак на ланцюжки постачання програмного забезпечення.

Розглянемо семантичні показники захищеності інформаційних систем.

Пропонується при проведенні інвестиційного аналізу засобів захисту інформації або для оцінки збитків у разі реалізації загроз враховувати збитки, як у вартісному обчисленні, так і нематеріальні збитки, завдані репутації, конкурентним можливостям суб'єкта господарювання.

Введемо семантичні показники «величина нематеріальних збитків» і «ймовірність завдання збитків», пов'язані з частотою реалізації загрози за конкретний період.

Семантичні показники «величина нематеріальних збитків» наведено в таблиці 4.7.

Таблиця 4.7 – Семантичні показники «величина нематеріальних збитків»

Величина збитку	Семантичний показник «величина нематеріальних збитків»
Мізерний	Збитком можна знехтувати.
Незначний	Витрати на ліквідацію наслідків реалізації загрози невеликі.
Помірний	Ліквідація наслідків реалізації загрози не пов'язана з великими витратами, але становище на ринку погіршується, частина клієнтів втрачається.

Продовження таблиці 4.7

Величина збитку	Семантичний показник «величина нематеріальних збитків»
Серйозний	Важко виконувати критично важливі завдання. Втрата на тривалий період становища на ринку. Ліквідація наслідків загрози пов'язана із значними фінансовими інвестиціями.
Критичний	Реалізація загрози призводить до неможливості вирішення критично важливих завдань. Організація припиняє існування.

В таблиці 4.7 можна побачити результат введення семантичного показника «величина нематеріальних збитків». Величина збитку може приймати такі значення: мізерний, незначний, помірний, серйозний та критичний.

Семантичні показники «ймовірність завдання збитків» наведено в таблиці 4.8.

Таблиця 4.8 – Семантичні показники «ймовірність завдання збитків»

Частота реалізації загрози	Значення ймовірності	Семантична характеристика реалізації загрози
Нульова	Близько нуля	Загроза практично ніколи не реалізується.
Один раз на декілька років	Дуже низька	Загроза реалізується рідко.
Один раз на рік	Низька	Скоріш за все, загроза не реалізується.
Один раз на місяць	Середня	Скоріш за все, загроза реалізується.
Один раз в неділю	Вище середньої	Загроза майже обов’язково реалізується.
Один раз на день	Висока	Шанси на позитивний результат вкрай малі.

В таблиці 4.8 можна побачити результат введення семантичного показника «ймовірність завдання збитків». При цьому частота реалізації загрози може

варіюватися від нульової до одного разу на день, а значення ймовірності реалізації загрози відповідно від близької до нуля до високої.

Пропонується здійснювати аналіз варіантів засобів захисту інформації за критерієм «вартість/ефективність», при цьому враховуючи міркування про те, що вартість засобів захисту інформації не повинна перевищувати певну суму. В даному випадку вартість засобів захисту не повинна бути більше 20 % вартості інформаційної системи.

Рівень шкоди не повинен перевищувати задане значення. В даному випадку рівень школи повинен бути не більше, ніж «незначний».

Надані оцінки відображають статичний стан об'єкта захисту, виходячи з наявних у засобах захисту інформації механізмів захисту, не враховуючи дійсну завантаженість методів захисту щодо нейтралізації наслідків атак, динаміку зміни загроз, можливість адаптації засобів захисту інформації до зміни загроз, не даючи рекомендацій щодо зміни складу механізмів захисту та структури засобів захисту інформації.

Розрахунок ризику проводиться з використанням значення загроз і цінності активу.

Значним недоліком існуючих методів оцінки ризиків у сфері розрахунку значення ризику інформаційної безпеки є оцінка вартості активів (розмір збитків) як умовних значень. Умовні значення не мають одиниць виміру, застосовних на практиці, зокрема, не є грошовим еквівалентом. У результаті це не дає реального уявлення рівня ризику, який можна перенести на реальні активи об'єкта захисту.

Таким чином, можна поділити процедуру розрахунку ризику на такі етапи:

– обчислення значення технічного ризику;
– обчислення потенційної шкоди.

Під технічним ризиком розуміється значення ризику інформаційної безпеки, що складається з ймовірностей реалізації загроз і використання вразливостей кожного компонента інформаційної системи з урахуванням рівня їх конфіденційності, цілісності та доступності.

Для обчислення значення технічного ризику можна навести наступні формули (4.1):

$$R_c = K_c \bullet P(A) \bullet P(B);$$
$$R_i = K_i \bullet P(A) \bullet P(B);$$

(4.1)

$$R_a = K_a \bullet P(A) \bullet P(B),$$

де R_c – значення ризику конфіденційності;

R_i – значення ризику цілісності;

R_a – значення ризику доступності;

K_c – коефіцієнт конфіденційності інформаційного активу;

K_i – коефіцієнт цілісності інформаційного активу;

K_a – коефіцієнт доступності інформаційного активу;

P(A) – можливість реалізації загрози;

P(B) – ймовірність використання вразливості.

Застосування даного алгоритму дозволить зробити детальнішу оцінку ризику, отримавши у результаті безрозмірне значення ймовірності виникнення ризику компрометації кожного інформаційного активу окремо.

Надалі можливе обчислення значення шкоди. Для цього використовується усереднене значення ризику кожного інформаційного активу та розмір потенційних втрат.

Значення збитків (L) розраховується за такою формулою (4.2):

$$L = R_{сер} \cdot S,$$

(4.2)

де $R_{сер}$ – середнє значення ризику;

S – втрати, ум. од.

Запропонована методика дозволяє коректно оцінити значення ризику інформаційної безпеки та скалькулювати грошові втрати у разі виникнення інцидентів безпеки.

Значення збитків (L) буде представлено в умовних одиницях.

Розробимо методику оцінки захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення. Дана методика буде складатися з наступних кроків.

1) Визначення методів захисту, які беруть участь в оцінці (табл. 4.5).

2) Ранжирування методів захисту (табл. 4.5) та розрахунок вагових коефіцієнтів цих методів за формулою (4.3).

3) Ранжирування видів кожного методу захисту (1 – технічні: програмні, криптографічні, апаратні та фізичні; 2 – комплексні; 3 – організаційні (правові); 4 – законодавчі) (табл. 4.5) та розрахунок вагових коефіцієнтів цих видів за формулою (4.6).

4) Відмітка про використання методів захисту кожного виду (табл. 4.5), які беруть участь в оцінці.

5) Формування матриці вагової функції по відміткам про використання.

6) Розрахунок оцінки для кожного методу захисту виду.

7) Розрахунок узагальнених показників по кожному методу захисту.

8) Розрахунок підсумкової оцінки.

9) Розрахунок оцінки, узагальнених показників по кожному методу захисту та підсумкової оцінки для методів захисту, які беруть участь в оцінці в ситуації, якщо було використано всі методи захисту кожного виду.

10) Порівняння результатів, що було отримано на кроках 8 та 9 та формування рекомендацій для підвищення захищеності інформаційної системи організації, що оцінюється.

Розглянемо детальніше методику оцінки захищеності інформаційної системи.

Обчислення вагових коефіцієнтів C_i для кожного і-го методу захисту для забезпечення безпеки виконується за формулою:

$$C_i = 1 - \frac{R_i - 1}{M},$$

(4.3)

де R_i– ранг;

M – число функціональних методів захисту.

Нормування коефіцієнтів, виконується наступним чином:

$$C_k = \frac{C_i}{\sum_{i=1}^{M} C_i}.$$

(4.4)

Ранжування видів методів захисту та обчислення вагового коефіцієнту виду відбувається за наступними формулами:

$$F_i = 1 - \frac{R_i - 1}{M},$$

(4.5)

$$F_k = \frac{F_i}{\sum_{i=1}^{M} F_i}.$$

(4.6)

Введемо вагову функцію, яка має 3 різні поведінки: «–1» – якщо метод захисту не було реалізовано, «0» – якщо метод захисту було реалізовано, але він не використовується на практиці та «1» – якщо метод захисту використовується. Ці значення заносяться у вагову функцію на етапі, коли експерт робить відмітки про використання методів захисту кожного виду (табл. 4.5), які беруть участь в оцінці.

В результаті анкетування методів захисту кожного виду обчислюється оцінка по виду:

$$W_{kj} = \sum_{k=1}^{n} F_k^j W_k,$$

(4.7)

де F_k^j – ваговий коефіцієнт k-виду j-методу захисту;

W_k– вагова функція k-виду.

Вагова функція виду складається з вагових функцій методів захисту цього виду.

Оцінка по кожному методу захисту обчислюється за наступною формулою:

$$X_j = \sum_{k=1}^{n} C_j W_{kj},$$

(4.8)

де C_j – ваговий коефіцієнт важливості j-го методу захисту;

W_{kj} – підсумковий показник по всім методам захисту k-виду.

Підсумкова оцінка захищеності визначається за формулою:

$$Y = \sum_j X_j.$$

(4.9)

В результаті можна отримати підсумкову оцінку захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення. В даному випадку цей показник буде відображати безпосередньо рівень захищеності ланцюжків постачання програмного забезпечення для умовної організації.

Проведемо розрахунок за розробленою методикою оцінки захищеності інформаційних систем.

В результаті підсумкова оцінка захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення, за такої умови, що всі методи захисту (табл. 4.5) було використано, складатиме 6,314 умовних одиниць. Дану загальну оцінку можна вважати ідеальною для даної інформаційної системи організації. В процентному еквіваленті оцінка захищеності складатиме 85 %, оскільки жодна інформаційна система не може бути захищена на 100 %.

За умови, якщо метод захисту, що представляє собою дотримання державної політики щодо вразливостей не було реалізовано, підсумкова оцінка захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення буде складати 5,905 умовних одиниць. Тобто, відсоток захищеності дорівнюватиме 79,49 %, що є на 5,51 % менше, ніж відсоток захищеності інформаційної системи організації за умови, що всі запропоновані методи захисту було використано.

Можна зробити висновки, що для максимально можливого для конкретної організації рівня захищеності ланцюжків постачання програмного забезпечення необхідно проконтролювати використання всіх запропонованих методів захисту.

Для підвищення захищеності інформаційної системи від атак на ланцюжки постачання програмного забезпечення доцільним буде застосування створеної

моделі захисту, що також включає методику забезпечення безпеки на кожному етапі розробки програмного забезпечення та управління ризиками безпеки ланцюжка постачання за фазою SDLC. Враховуючи те, що кожен крок ланцюжка постачання програмного забезпечення є вразливим до атак, його слід перевіряти на кожному кроці процесу, включаючи процес розробки програмного забезпечення.

4.4 Оцінка ефективності запропонованих методів

Створена модель захисту ланцюжків постачання програмного забезпечення може збільшити захищеність інформаційної системи умовної організації від зловмисного впливу на 15 %, якщо порівнювати з ІС, для якої дана модель захисту не використовується.

Методи та засоби захисту, що були наведені допоможуть значно мінімізувати ризики атак та забезпечити захищеність ланцюжків постачання програмного забезпечення для організації. В результаті реалізації даних методів захищеність інформаційної системи організації складатиме 80 %.

Для розрахунку точного значення захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення можна скористатися запропонованою методикою оцінки захищеності інформаційних систем.

Для інформаційної системи, кількість методів захисту якої складає всі 24 запропоновані елементи (табл. 4.5), було визначено, що значення підсумкової оцінки захищеності буде дорівнювати 6,314 умов. одиниць або 85 % в процентному еквіваленті. Дану загальну оцінку можна вважати ідеальною для інформаційної системи організації, тому що в цьому випадку всі методи захисту було реалізовано, та враховуючи той факт, що жодна інформаційна система не може бути захищена на 100 %.

Отже, створення моделі захисту ланцюжків постачання програмного забезпечення та реалізація методики оцінки їх захищеності можуть бути корисними для організацій, які хочуть відстежувати та забезпечувати безпеку своїх ланцюжків постачання.

Реалізація методів та засобів захисту ланцюжка постачання програмного забезпечення повинна бути ключовим завданням для будь-якої організації.

ВИСНОВКИ

В кваліфікаційній роботі було вирішено задачу щодо розробки моделі захисту ланцюжків постачання програмного забезпечення.

З цією метою було розглянуто типи атак на ланцюжки постачання програмного забезпечення; методи пом'якшення загроз ланцюжка постачання; методи безпеки, які слід враховувати командам безпеки; розглянуто ланцюжок постачання в NIS2 Directive та цикл PDCA для кібербезпеки ланцюжка постачання; статистику атак на ланцюжок постачання програмного забезпечення, останні відомі атаки даного типу та методи зменшення ризиків безпеки; життєвий цикл безпечної розробки програмного забезпечення (SSDLC) тощо.

Було опрацьовано такі питання, як аналіз технологій перевірки та аналіз атак на ланцюжки постачання програмного забезпечення, проведено дослідження методів забезпечення безпеки на кожному етапі розробки програмного забезпечення.

Ланцюжок постачання програмного забезпечення складається з компонентів, бібліотек, інструментів і процесів, які використовуються для розробки, створення та публікації артефактів програмного забезпечення. Як відомо, постачальники програмного забезпечення часто створюють продукти, збираючи компоненти комерційного програмного забезпечення з відкритим кодом.

Як вже зазначалося, сьогодні суть розробки програмного забезпечення полягає в тому, що постачальники та розробники програмного забезпечення часто створюють продукти, збираючи компоненти комерційного програмного забезпечення з відкритим кодом, проводять його тестування та якнайшвидше розгортають його у робочому середовищі. У той час як швидкість розгортання є ключовим пріоритетом, безпека ланцюжка постачання програмного забезпечення є не менш важливою задачею для будь-якої організації.

Атаки на ланцюжки постачання – це загрози, що націлені саме на розробників і постачальників програмного забезпечення. Головна мета таких загроз полягає в отриманні доступу до вихідних кодів, створенні процесів або механізмів оновлення шляхом зараження легальних програм для поширення шкідливого програмного забезпечення. І дуже часто це означає використання відкритих і пропрієтарних компонентів, запозичених з різних джерел.

Зловмисники використовують сучасні гіперз’єднані ланцюжки постачання програмного забезпечення, оскільки один «пролом» дозволяє їм отримати доступ до великої кількості систем, що знаходяться нижче. Це означає, що команди безпеки та організації-розробники у всьому світі повинні ставити безпеку ланцюжка постачання на перше місце.

Саме тому безпека ланцюжка постачання програмного забезпечення має вирішальне значення. Вона повинна поєднувати в собі найкращі практики управління ризиками та кібербезпеки, допомагаючи захистити ланцюжок постачання програмного забезпечення від потенційних вразливостей.

Передова практика забезпечення кібербезпеки ланцюжків постачання повинна охоплювати декілька областей, а саме: стратегічний корпоративний підхід; управління ризиками ланцюжка постачання; управління взаємовідносинами із постачальниками; обробку вразливостей; перевірку якості продукції та практик постачальників та провайдерів послуг.

В результаті було розроблено модель захисту ланцюжків постачання програмного забезпечення, що включала методику забезпечення безпеки на кожному етапі розробки програмного забезпечення та управління ризиками безпеки ланцюжка постачання за фазою SDLC. Створено методику оцінки захищеності інформаційної системи організації від атак на ланцюжки постачання програмного забезпечення та проведено оцінку ефективності розробленої моделі та методики.

Окремі результати роботи було опубліковано в статті та тезах доповідей на Міжнародних наукових конференціях [1 – 3].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Новікова Д. Дослідження методів забезпечення безпеки в процесі розробки програмного забезпечення. *Наукові відкриття та фундаментальні наукові дослідження: світовий досвід* : матеріали III міжнар. наук. конф., м. Вінниця, 24 лист. 2023 р. Вінниця, 2023. С. 331 – 336.
2. Новікова Д. Типи атак та методи пом'якшення загроз ланцюжка поставок програмного забезпечення. *Актуальні питання розвитку галузей науки* : матеріали II міжнар. наук. конф., м. Чернігів, 01 груд. 2023 р. Чернігів, 2023. С. 260 – 262.
3. Новікова Д. Зменшення ризиків безпеки при керуванні ланцюжком постачання програмного забезпечення. *Здобутки та досягнення прикладних та фундаментальних наук XXI століття* : матеріали VI міжнар. наук. конф., м. Черкаси, 11 груд. 2023 р. Черкаси, 2023. С. 244 – 246.
4. Papaphilippou M., Moulinos K., Theocharidou M. Good practices for supply chain cybersecurity. *European Union Agency for Cybersecurity (ENISA)*. 2023. P. 3 – 7.
5. Новікова Д. О. Аналіз атак на ланцюжки поставок програмного забезпечення : квал.роб / Харків. ун-т радіоелектр. Харків, 2022. 58 с.
6. Geer D., Tozer B., Meyers J. S. Counting Broken Links: A Quant’s View of Software Supply Chain Security. *Usenix*. 2020. Vol. 45, No 4. P. 1 – 4.
7. What is software supply chain security? : веб-сайт. URL: https://www.redhat.com/en/topics/security/what-is-software-supply-chain-security (дата звернення: 05.09.2023).
8. Threat landscape for supply chain attacks : веб-сайт. URL: https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks (дата звернення: 06.09.2023).
9. Directive – 2022/2555 – EN – EUR-Lex : веб-сайт. URL: https://eur-lex.europa.eu/eli/dir/2022/2555 (дата звернення: 06.09.2023).
10. Global Cybersecurity Outlook 2022 : веб-сайт. URL: https://www.weforum.org/reports/global-cybersecurity-outlook-2022/ (дата звернення: 07.09.2023).

11. 2022 Security Trends: Software Supply Chain : веб-сайт. URL: https://anchore.com/blog/2022-security-trends-software-supply-chain-survey/ (дата звернення: 07.09.2023).
12. M-Trends 2022: Cyber Security Metrics, Insights and Guidance From the Frontlines : веб-сайт. URL: https://www.mandiant.com/resources/m-trends-2022 (дата звернення: 07.09.2023).
13. Lella I., Tsekmezoglou E., Svetozarov R. N., Ciobanu C., Malatras A., Theocharidou M. ENISA Threat Landscape 2022 report. *European Union Agency for Cybersecurity*. 2022. URL: https://www.enisa.europa.eu/news/volatile-geopolitics-shake-the-trends-of-the-2022-cybersecurity-threat-landscape (дата звернення: 08.09.2023).
14. 2022 Global Digital Trust Insights Survey : веб-сайт. URL: https://www.pwc.com/gx/en/issues/cybersecurity/global-digital-trust-insights.html (дата звернення: 08.09.2023).
15. Top 5 Famous Software Supply Chain Cyber Attacks : веб-сайт. URL: https://www.cloudsek.com/blog/top-5-famous-software-supply-chain-cyber-attacks-in-2023 (дата звернення: 08.09.2023).
16. NIS 2 Directive Text, Article 19 : веб-сайт. URL: https://www.nis-2-directive.com/NIS_2_Directive_Article_19_(Proposal_16.12.2020).html (дата звернення: 08.09.2023).
17. The NIS 2 Directive : веб-сайт. URL: www.nis-2-directive.com (дата звернення: 09.09.2023).
18. Software Supply Chain Attacks : веб-сайт. URL: https://www.aquasec.com/cloud-native-academy/supply-chain-security/software-supply-chain-attacks/ (дата звернення: 09.09.2023).
19. Software Supply Chain Attacks. *National Counterintelligence and Security Center (NCSC)*. 2021. URL: https://www.dni.gov/files/NCSC/documents/supplychain/Software_Supply_Chain_Attacks.pdf (дата звернення: 09.09.2023).
20. Secure SDLC | Secure Software Development : веб-сайт. URL: https://snyk.io/learn/secure-sdlc/ (дата звернення: 10.09.2023).
21. Enhancing SSDLC with OWASP SAMM: A Comprehensive Guide : веб-сайт. URL: https://codific.com/secure-software-development-lifecycle/ (дата звернення: 10.09.2023).
22. Ellison R. J. Evaluating and Mitigating Software Supply Chain Security Risks. *Technical Note CMU/SEI-2010-TN-016. Software Engineering Institute.*

2010. URL: www.sei.cmu.edu/library/abstracts/reports/10tn016.cfm (дата звернення: 01.10.2023).

23. Oehmen J., Ben-Daya M., Khan O. Integrating Supply Chain Risks in Product Development: A Conceptual Framework. *Proceedings of the Tenth International Research Seminar on Supply Chain Risk Management*. *ISCRiM*. 2010. P. 56 – 61. URL: www.husdal.com/wp-content/uploads/2010/09/Proceedings-ISCRiM-2010.pdf (дата звернення: 01.10.2023).

24. Goertzel K. M. Supply Chain Risk Management and the Software Supply Chain. *OWASP AppSec DC*. 2010. URL: https://www.owasp.org/images/7/77/BoozAllen-AppSecDC2010-sw_scrm.pdf (дата звернення: 02.10.2023).

25. SQL Injection Prevention Cheat Sheet : веб-сайт. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 03.10.2023).