МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛИТВИНЕНКО СЕРГІЙ КОСТЯНТИНОВИЧ

Допускається до захисту:
в.о завідувача кафедри
інформаційних технологій,
_____ Нескородєва Т. В
«____» _____ 20__ р.

ОЦІНКА РИЗИКІВ API ВЕБ-СЕРВІСІВ

Спеціальність 125 Кібербезпека

Бакалаврська робота

Науковий керівник:
Барибін О.І., доцент кафедри
радіофізики та кібербезпеки
к.т.н

Оцінка:___/ ___/ _____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця — 2021

Литвиненко С. К. ОЦІНКА РИЗИКІВ API ВЕБ-СЕРВІСІВ. Спеціальність 125 Кібербезпека. Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній роботі здійснено оцінку ризиків інформаційної для API веб сервісів відповідно до методології оцінки ризиків OWASP Risk Rating Methodology. Результатом роботи є реєстр ризиків, що містить у собі 10 найменувань ризиків.

Ключові слова: ІНФОРМАЦІЙНА БЕЗПЕКА, ОЦІНКА РИЗИКІВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ, OWASP RISK RATING METHODOLOGY, РИЗИКИ API ВЕБ СЕРВІСІВ.

Табл. 25, Рис. 2, Бібліограф. 17 найм.

Lytvynenko S. K. RISK ASSESSMENT OF API WEB SERVICES. Specialty 125 Cybersecurity. Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In the qualification work, the risk assessment of information for API websites was carried out in accordance with the OWASP Risk Rating Methodology. The result is a risk register, which contains 10 names of risks.

Keywords: INFORMATION SECURITY, INFORMATION SECURITY RISK ASSESSMENT, OWASP RISK RATING METHODOLOGY, API WEB SERVICES RISKS.

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ ПРО API 6
1.1 Знайомство з API 6
1.2 Повсякденні приклади API 8
1.3 Чому сучасні сайти електронної комерції використовують API 9
РОЗДІЛ 2. РИЗИКИ API ТА ЇХ ОЦІНКА 12
2.1 Баланс між безпекою та легкістю використання 12
2.2 Огляд методолії оцінки ризиків OWASP 13
2.3 Ідентифікація ризиків 14
2.4 Фактори для оцінки ймовірності 17
2.5 Фактори для оцінки впливу 25
2.6 Визначення серйозності ризику 34
2.7 Заходи для зменшення ризиків 40
ВИСНОВКИ 46
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 47

ВСТУП

Прикладні програмні інтерфейси (API) більше не є просто інструментом розробника. Вони є основою широкого спектру бізнес-додатків та веб-сервісів. API з'єднують програми всередині компаній та обмінюються даними між організаціями. [1]

API стали засобом доступу до даних за допомогою цифрових каналів, таких як мобільні додатки, хмарні технології та Інтернет речей, що полегшує підприємствам надання доступу до своєї інформації для широкої аудиторії, будь то клієнти, партнери чи співробітники. Однак, оскільки API стають частиною стандартної корпоративної архітектури, їх профіль ризику безпеки постає проблемою, яка потенційно зменшує привабливість цієї потужної технології інтеграції. Зловмисні напади та атаки на відмову в обслуговуванні (DoS) все частіше орієнтуються на корпоративні програми, оскільки внутрішні системи стають більш доступними та придатними для використання за допомогою хмарних, мобільних та локальних середовищ. API може стати основною точкою вразливості, враховуючи його здатність пропонувати програмний доступ зовнішнім розробникам.

Отже, безпека є важливим елементом стратегії API будь-якої організації. Хоча безпека API поділяє багато аспектів, спільних як для безпеки веб-сайтів, так і для мережевої безпеки, вона також принципово відрізняється як з точки зору використання, так і унікальних областей додаткових ризиків, до яких сприйнятливі API. Наприклад, API переміщують межі взаємодії від веб-рівня до серверних програм та джерел даних безпосередньо. [2]

Проблеми безпеки API є природним спадкоємцем попередніх хвиль проблем безпеки в Інтернеті. Коли підприємства вперше підключилися до Інтернету на початку 1990-х, вони зіткнулися з попередником сучасних хакерів: зловмисних користувачів, які перевіряли комп'ютери на наявність відкритих портів та вразливостей платформи. Щоб запобігти порушенням, організації розгорнули брандмауери та системи запобігання вторгненню (IPS). Однак, коли ці самі організації відкрили доступ до своїх веб-додатків, хакери

швидко обійшли брандмауери, і вони використовували такі методи, як кодування та коментарі, щоб уникнути виявлення підпису IPS.

Брандмауери веб-додатків, які вийшли на сцену через кілька років, могли визначити тип трафіку додатків, наприклад HTTP або обмін миттєвими повідомленнями. Але, на жаль, ця обізнаність у програмі дає мало переваг проти запобігання атакам API. Брандмауери наступного покоління не можуть блокувати атаки, які використовують специфічні вразливості API. Вони не можуть виявити різні види атак JSON або XML, а також атак фальсифікації параметрів. Вони не можуть зупинити шахрайські пристрої або атаки бізнес-логіки. Організації, які покладаються виключно на рішення мережевої безпеки для захисту своїх API, не повинні дивуватися, якщо вони зазнають порушення програм API.

Метою даної кваліфікаційної роботи є оцінка ризиків API веб сервісів за методологією OWASP Risk Rating Methodology та розробка рекомендацій із зниження оцінюваних ризиків.

Завдання дипломної роботи:

1. Дослідити ключові фактори технологій API з точки зору інформаційної безпеки;

2. Скласти перелік найбільших ризиків API та оцінити їх;

3. Розробити рекомендації для зниження ризиків API - сервісів.

Об'єктом дослідження є прикладний програмний інтерфейс (API).

Предметом дослідження є ризики інформаційної безпеки використання API.

РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ ПРО API

1.1 Знайомство з API

API (прикладні програмні інтерфейси) - це набір функцій, що дозволяє програмам отримувати доступ до даних та взаємодіяти із зовнішніми програмними компонентами, операційними системами або мікросервісами.

Для спрощення, API надає відповідь користувача системі та надсилає відповідь системи назад користувачеві. Ви натискаєте "додати в кошик;" API повідомляє сайту, що ви додали товар у кошик; веб-сайт розміщує товар у вашому кошику, і ваш кошик оновлюється. Ви можете почути термін "мікросервіс", що стосується API. Однак це не одне і те ж. Мікросервіси - це стиль або архітектура, що розділяє функціональність всередині веб-програми. Хоча API - це структура, яка розробниками взаємодіє з веб-додатком. Мікросервіси насправді можуть використовувати API для взаємодії між собою. API дозволяє розробнику зробити конкретний «дзвінок» або «запит» для надсилання або отримання інформації. Цей зв’язок здійснюється за допомогою мови програмування під назвою “JSON”. Він також може бути використаний для здійснення визначеної дії, наприклад оновлення або видалення даних. Існує чотири основних методи запиту, які можна створити за допомогою API:

GET - Збирає інформацію (витягуючи всі коди купонів)

PUT - Оновлення фрагментів даних (оновлення цін на товари)

POST - Створює (Створення нової категорії товару) ВИДАЛИТИ - (Видалення публікації в блозі)

Отже, що таке JSON і чому він використовується? JSON (JavaScript Object Notation) використовується для представлення даних на сервері. Це досить просто для читання людьми та легке для розуміння машинами / програмами. Давайте подивимось на приклад JSON із продукту на BigCommerce: Це легко зрозуміти, оскільки воно виводиться в парах ключ / значення з ключем ліворуч і значенням праворуч. Ключі є фіксованим

об'єктом, визначеним додатком, і залишатимуться тими ж, що і з "категорією". Тоді як значення будуть унікальними, наприклад, «Сорочки».

Існує кілька компонентів API-запиту, за рахунок яких він функціонує.

Заголовки надають інформацію клієнту та серверу. Поширеними прикладами заголовка можуть бути автентифікаційні дані, такі як "Маркер авторизації" або "Ідентифікатор клієнта". Ці облікові дані надаються вам автоматично під час створення облікового запису API. Інший поширений заголовок називається "Тип вмісту", який інформує сервер про те, який тип контенту буде надіслано. Наприклад, загальновживаним типом вмісту є "application / json", який повідомляє сервер, ми надсилаємо дані JSON.

Методи - це дії, що здійснюються під час надсилання запиту. Подумайте про початок, коли ми обговорювали GET, PUT, POST та DELETE. Це всі методи API.

Дані запиту, які також називають "тілом", - це інформація, яка буде надіслана або повернута сервером. У попередньому обговоренні JSON ви можете побачити приклад даних API. Текст запиту іноді вимагає конкретної інформації, перш ніж його можна буде доставити. Прикладом цього є те, що якщо ви редагуєте окремий продукт, перед внесенням будь-яких змін потрібно буде ввести ідентифікатор продукту.

Як щодо REST та SOAP API? Хоча API дотримується певного набору правил, що визначають спосіб взаємодії програм між собою. REST & SOAP визначають спосіб представлення API. Кожен з них подібний за функціональністю, але має кілька ключових відмінностей та випадків використання. REST (скорочення від Representational State Transfer) - це визначена архітектура програмного забезпечення для веб-розробки, яка зазвичай використовується для взаємодії HTTP. RESTful API (або просто REST API) - це запрошення від клієнтського ПО до серверного ПО, які слідують принципам REST, дозволяючи веб-клієнтам та серверам взаємодіяти з величезним різноманіттям веб-ресурсів. API REST використовує стандартні методи HTTP та коди, що забезпечуються для забезпечення певного рівня

стандартизації. [3] Доступ до нього здійснюється через HTTP-URL і широко використовується для веб-сервісів і він являє собою набір правил, яких розробники дотримуються при створенні API. REST читається за допомогою JSON, які ми розглядали раніше. Одне з цих правил полягає в тому, що API повинен бути розроблений таким чином, щоб він був простим у використанні та мав сенс для розробників. Прикладом недотримання цього правила може бути кінцева точка продукту „prod_839” замість просто „products”. Оскільки це може спричинити неприємну роботу з API.

SOAP або Простий протокол доступу до об’єктів - ще один спосіб розробки веб-служб. Замість типового JSON, який використовує REST API. SOAP використовує мову, відому як Extensible Markup Language (XML). XML розроблений таким чином, щоб його можна було прочитати машиною та людиною. SOAP дотримується суворого стандарту правил, таких як структура обміну повідомленнями та домовленості щодо надання запитів або відповідей. В основному в сучасному світі використовується REST API. Такі компанії, як Google, Amazon, eBay і навіть BigCommerce, використовують REST API. REST, як правило, є кращим вибором серед розробників програмного забезпечення.

1.2 Повсякденні приклади API

API допомагає розробникам швидко доставляти інформацію споживачам і використовується щодня в сучасному світі. Від покупок в Інтернеті, перегляду додатків у соціальних мережах або відтворення гри на своєму смартфоні. Щоразу, коли ви відвідуєте сторінку в Інтернеті, ви взаємодієте з API. Ось кілька реальних прикладів того, як ви взаємодієте з API, і, можливо, навіть не усвідомлюєте. Звернення в банк. Якщо подумати про себе як про користувача, а про касира як про API, тоді як банк - це система, з якою ви взаємодієте. Коли ви хочете забрати гроші зі свого рахунку, ви підходите до каси (API) і говорите: "Я хочу отримати 200 гривень з цього рахунку". Потім касир (API) переходить на задню панель і каже менеджеру

банку (системі) "Пан / пані X 200 гривень ", менеджер банку (система) видає касиру (API) 200 гривень, який в кінцевому підсумку видає вам . Як ви можете бачити API - це месенджер між вашими потребами та системою. При пошуку готелів, коли ви заходите на туристичний сайт, він може бути пов’язаний з 10 іншими туристичними сайтами, щоб знайти найкращу пропозицію для вас. Коли ви вводите такі дані, як Київ, 2 ночі, 1 кімната, ви надсилаєте цей запит тим 10 різним сайтам. API приймає ваш запит для цього конкретного місця, діапазону дат, приміщення та відсилає 10 сайтів, які повертають знайдені угоди. Ви переглядаєте 10 пропозицій і вибираєте найкращу.

1.3 Чому сучасні сайти електронної комерції використовують API

API пропонує широкий спектр переваг для сайтів електронної комерції. Вони можуть допомогти споживачам легко знаходити товари, розвивати бренд компанії або навіть розширити свій потенціал заробітку, продаючи товари на різних ринках, таких як eBay, Amazon та Facebook. Нижче наведено деякі переваги того, чому API настільки важливий для веб-сайтів електронної комерції сьогодні.

Безпека. Безпека підвищується, коли сайти використовують API. Щоразу, коли ви надсилаєте запит, ви не маєте прямого зв’язку з сервером. Ви надсилаєте невелику кількість інформації, API доставляє її, а сервер відправляє її назад. Це мінімізує ризик порушення або доступу когось до серверної частини.

Швидкість. Без API, вам доведеться зателефонувати в магазин і попросити їх переглянути їх інвентар від усіх своїх постачальників, який вони зрештою отримають до вас. Це замість того, щоб мати API, де ви могли легко побачити, що це за продукт, ціна чи рівень запасу.

Масштабованість. API дозволяють масштабовано та гнучко розширювати каталог вашого магазину, безпеку чи потреби у даних. Ваш магазин може рости швидше, коли вам не потрібно враховувати новий код для кожного товару чи користувача.

Додатки на основі API значно відрізняються від традиційних програм. Раніше, наприклад, користувачі / відвідувачі мали доступ до веб-сервера через браузер, і більша частина "обробки даних" проводилася на самому сервері. Оскільки клієнтські пристрої ставали все різноманітнішими та дедалі потужнішими - із швидшими процесорами, великою пам’яттю, більшою пропускною здатністю тощо - більша частина логіки відійшла від виконання на внутрішніх серверах до інтерфейсу (тобто на самому клієнтському пристрої), як зазначено на графіку нижче. [4]

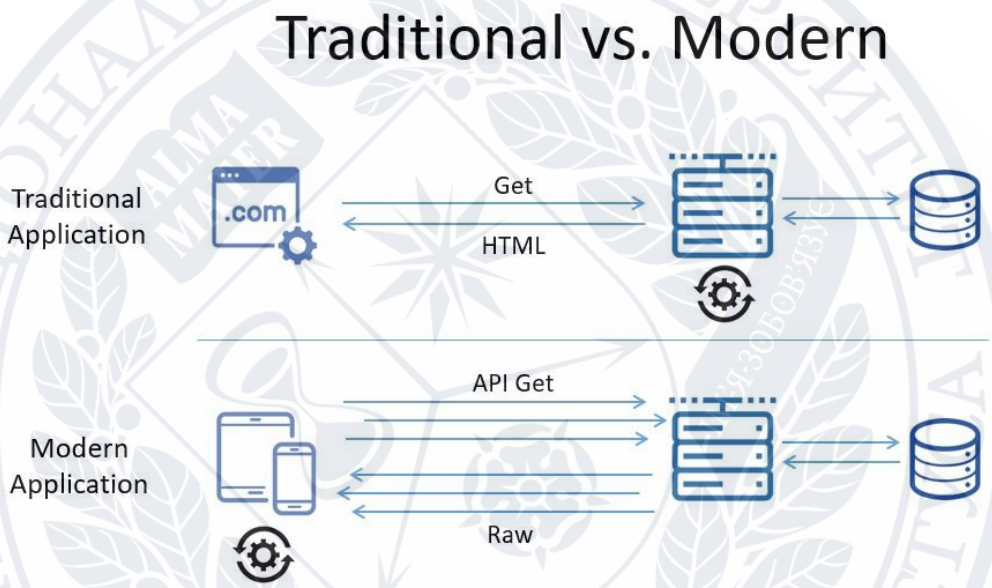


Рисунок 1 – методи побудови додатків

1.4 Особливості забезпечення безпеки API

1. Безпека API є невід’ємною частиною API. Досить часто користувачі розглядають безпеку API як їх особливість. Це не особливість. Це інша технологія. Для захисту вашого API потрібно шукати рішення деінде, окрім самого API. Безпека API - це спосіб мислення, а не особливість.

2. Захист API на основі програмного забезпечення - це опція, доступна вам під час управління API. Це досить зручно, і якщо у вас і так багато розуміння того, як все це працює, ви можете подумати, що все добре. На жаль,

ви помилитеся, і є історії, які показують, чому. Усі сумнозвісні порушення безпеки API були пов’язані з програмним забезпеченням: запуск чужорідного коду на вашому сайті призведе до цілого ряду вразливостей. Отже, виберіть більш конкретний варіант, такий як тестування безпеки API.

3. Сам концепт API можна підсумувати з достатньою мірою простоти, так: дві програми пов’язані через API, і все. Однак безпека API не така проста. І це може бути однією з причин для того, щоб задуматися про інвестування в якісь зовнішні поради від експерта в цій галузі. Іронічним є те, що чим простішим є ваше фактичне з’єднання з API, тим менш простим буде його захист. У сучасну епоху спільний доступ до даних, але в той же час їх захист - це те, що робить безпеку API обов’язково складним завданням.

4. Шлюзи безпеки API слід постійно використовувати як вирішення поточних проблем безпеки API. Шлюзи безпеки здатні обмежити потік даних саме тими даними, які вам потрібно передати, і обмежити передавання даних, які не обов’язково мають передаватись. Звичайний шлюз API може бути корисним для вашого з'єднання, але він ніколи не зрівняється із захищеним шлюзом API.

5. Багато компаній говорять про безпеку своїх продуктів, оскільки вони мають особливості безпеки API, і вони вважають, що безпека API в кінцевому рахунку означає найкращу безпеку. Однак це неправда. Наявність функцій безпеки API не означає, що ваш продукт захищений або безпечніший за інші. Незважаючи на те, що API можуть покращити вашу безпеку та захист, а також посилити захист, яку надає ваша система безпеки, вони не гарантують належний рівень безпеки самі по собі. Це потрібно знати, оскільки люди дуже часто вірять, що самої безпеки ідентифікаційних даних API може бути достатньо.

РОЗДІЛ 2. РИЗИКИ API ТА ЇХ ОЦІНКА

2.1 Баланс між безпекою та легкістю використання

API надають дані або функціональні можливості для використання програмами та розробниками, а це означає, що саме вони є дверима та вікнами, що дозволяють отримати доступ до цінних цифрових активів бізнесу - і, отже, до самого серця бізнесу. Як і всі двері та вікна, що забезпечують доступ до чогось цінного, API повинні бути розроблені з найвищим рівнем безпеки. Розробники API загалом розуміють важливість дотримання принципів проектування API, оскільки ніхто не хоче розробляти або впроваджувати поганий API. Незважаючи на це, може бути спокусливо шукати прості шляхи, щоб відповідати агресивним термінам спринту, дістатися до фінішу та розгорнути API. Хоча це і зрозуміло, ці шляхи можуть становити серйозний ризик - незахищені API. Вразливі API можуть піддавати основні дані та послуги бізнесу різноманітним шкідливим атакам.

Багато великих підприємств та організацій зазнали порушень, наслідки яких можуть бути від соромних до катастрофічних, що є результатом дір в захисті API. Ці отвори не завжди виникають через недбалість. Розробка безпечних API не так проста, як може здатися, оскільки постачальник API часто повинен знаходити баланс між простотою використання та безпекою.

Однією з головних цілей API є простота використання; API не має великої цінності, якщо розробники його не використовують, і ніхто не хоче працювати з API, який настільки заблокований механізмами безпеки, що заважає продуктивності. Проблема безпеки API - це не блокування API; скоріше, це управління API, які є безпечними, але при цьому все одно гнучкими та досить доступними для сприяння інноваціям. Порушення цього балансу означає, що постачальникам API, як правило, слід уникати складних системних залежностей та жорстких моделей управління, які були типовими для попередніх поколінь IT-стратегії, коли підключені цифрові ринки ще не досягли популярності, а корпоративний брандмауер розглядався як фортеця

для відсічі сторонніх . Справді, застарілі методи безпеки в цілому можуть бути нежиттєздатними, оскільки вони покладаються на обмеження швидкості, з якою підприємства оновлюють своє програмне забезпечення. Підприємства, які наполягають на повільному, громіздкому процесі оновлення, ризикують не реагувати на нові уразливі місця. Для більшості організацій найкращим шляхом є прийняття технологічних стратегій, таких як розробка API, що підтримують постійні оновлення та забезпечують можливість впровадження засобів контролю безпеки. Як частина продуктового мислення API, підприємствам слід завжди враховувати наступні найкращі практики безпеки API: шифрування даних, авторизація кінцевого користувача та програми, обмеження швидкості та виявлення ботів. [5]

2.2 Огляд методології оцінки ризиків OWASP

Із допомогою методології OWASP Risk Rating Methodology можна оцінити серйозність всіх виявлених ризиків для бізнесу і прийняти обґрунтоване рішення про те, що робити з цими ризиками. Наявність системи для рейтингових ризиків заощадить час і усуне суперечки про пріоритети. Ця система допомагає забезпечити, щоб бізнес не відволікався на незначні ризики, ігноруючи при цьому більш серйозні ризики, які менш добре вивчені.

Основні поняття методології:

Ризик - це небезпека, наслідки, які можуть виникнути в результаті поточного процесу чи майбутніх подій.

Управління ризиками - це процес управління, який охоплює ідентифікацію, оцінку та контроль ризику, який може загрожувати безперервності бізнесу або діяльності компанії.

Загальні цілі: зменшити витрати, запобігти збиттю компаній, збільшити корпоративний прибуток, зменшити виробничі витрати та багато іншого.

Оцінка ризику - це методи, що проводяться для визначення того, чи є діяльність / ризик прийнятним чи ні.

Хорошу оцінку повинна робити кваліфікована та досвідчена команда.

Кожна компанія чи організація мають різні рівні прийнятності.

Ризик = ймовірність * вплив

Як користуватися методологією оцінки ризику OWASP:

Крок 1: Ідентифікація ризику

Крок 2: Фактори для оцінки ймовірності

Крок 3: Фактори для оцінки впливу

Крок 4: Визначення серйозності ризику [6]

2.3 Ідентифікація ризиків

Порушений дозвіл на рівні об’єкта. "Конфіденційні дані знаходяться у загальнодоступному сегменті", - стверджується в заголовку щотижня, повідомляючи про чергове порушення даних. Це ілюструє ризик №1 для безпеки веб-API: неправильний або відсутність контролю доступу до ресурсів, доступних з Інтернету. Це може означати, що авторизація взагалі не була реалізована або що вона впроваджена, але не використовується. Залежно від типу активу, наслідки можуть бути від розкриття даних, втрати чи маніпуляцій до повного поглинання рахунку. Як правило, легко використати цю вразливість, як тільки вона виявлена. Все, що зловмиснику потрібно зробити, це змінити ідентифікатор у запиті, і він потенційно отримав доступ до об’єктів, до яких їм не дозволяється. [7]

Наприклад, ми маємо посилання такого типу ‘https://www.datastoresite.com/data/111’ .Цей виклик API використовується для отримання приватних даних користувача. Використовуваний ресурс - це дані. Ми маємо ідентифікатор повідомлення - 111. Це важлива частина для зловмисника. Ідентифікатор повідомляє службі, який запис повернути. API отримує цей запис із сховища даних і повертає його у відповідь. Якщо дані з ідентифікатором 111 були призначені для нашого користувача, ми повинні мати можливість їх отримати. Але якщо дані належали іншому користувачеві, API ніколи не повинен їх повертати. Якщо нам вдалося таким чином отримати

дані іншого користувача, то у нас сталася помилка авторизації об’єкта Broken Object Level Authorization.

Порушена автентифікація користувача. Завжди складна проблема, автентифікація користувачів стає ще більш плутаною з API через різноманітність агентів, які можуть отримати доступ до кінцевої точки. Гірше того, кінцева точка автентифікації має бути доступною кожному, що робить її резонансною ціллю. Зловмисники можуть отримати доступ до облікових записів користувачів для отримання конфіденційних даних, видавання себе за користувачів, викрадення грошей або заволодіння обліковим записом. [8]

Надмірна експозиція даних. Деякі API можуть бути реалізовані до того, як буде доступна точна специфікація. Це призводить до загальних реалізацій, де API надає всі властивості об'єкта, а клієнт API повинен використовувати лише те, що потрібно. Це може призвести до виявлення конфіденційних даних, включаючи особисту інформацію. Одним з прикладів може бути кінцева точка API, яка завжди повертає повний об’єкт користувача JSON, навіть якщо клієнтській програмі потрібна лише кілька атрибутів. Як запобігти: Кінцеві точки API повинні реалізовувати фільтрацію даних, щоб повертати лише ті властивості, які потрібні для конкретного випадку використання. Тестування повинно включати перевірку всіх відповідей API, щоб запобігти надмірному впливу. [9]

Брак ресурсів та обмеження тарифів. Конфігурації за замовчуванням для багатьох загальнодоступних API не обмежують поведінку клієнтів API з точки зору швидкості доступу чи кількості та розміру запитуваних ресурсів. Однією з причин цього (крім недогляду) може бути те, що на етапі проектування не були відомі відповідні межі для конкретного випадку використання. Експлуатація може призвести до вичерпання системних ресурсів і, як наслідок, відмови в обслуговуванні (DoS). Така сама поведінка може бути використана для заповнення облікових даних та атак грубої сили або використана для активації помилок переповнення. [10]

Порушена авторизація на рівні функції. Кожен виклик функції API повинен вимагати авторизацію, щоб гарантувати, що поточний користувач має право виконувати певну функцію на конкретному об'єкті. Якщо це неправильно реалізовано, користувачі можуть мати можливість викликати функції, до яких вони не повинні мати доступу, наприклад, вручну змінюючи імена функцій в URL-адресах або методи HTTP у запитах. Такі вади можуть виявити надмірну функціональність, оскільки зловмисники найчастіше націлюються на адміністративні функції. [11]

Масове призначення. Деякі виклики API вимагають або змінюють більше одної властивості, і всі параметри запиту автоматично використовуються як властивості об'єкта. Якщо їх не обмежити належним чином, кінцеві точки API можуть надавати доступ до властивостей, що перевищують дозволений обсяг, дозволяючи зловмисникам вказувати додаткові параметри для модифікації об’єктів. Наприклад, якщо значення балансу облікового запису користувача передається в інформації профілю користувача, зловмисник може надіслати запит, який змінює баланс рахунку.

Неправильна конфігурація безпеки. Це загальноприйнята категорія для всіляких проблем у всьому стеці технологій, які можуть викрити конфіденційну інформацію або забезпечити плацдарм для подальших атак. Це може включати використання конфігурацій за замовчуванням без автентифікації або слабкої автентифікації, неможливість застосування HTTPS, залишення непотрібних методів HTTP на місці, опускання важливих заголовків безпеки HTTP тощо. [12]

Ін'єкція. Зловмисники можуть намагатися подавати шкідливі дані до кінцевої точки API, сподіваючись, що вони будуть надіслані безпосередньо інтерпретатору та виконані як код. Без належної перевірки будь-який вхід API може надавати вектор введення. Застосовуються всі типові атаки введення веб-додатків, включаючи введення команд ОС та ін'єкцію SQL, NoSQL та LDAP. Наслідки варіюються від розкриття інформації до відмови в обслуговуванні та віддаленого виконання коду.

Неправильне управління активами. Порівняно з веб-додатками, заснованими на користувальницькому інтерфейсі, веб-інтерфейси часто виявляють набагато більше кінцевих точок, залишаючись менш помітними, особливо для децентралізованих розгортань, розподілених між різними службами чи мікросервісами. Неповні або застарілі запаси активів та документація можуть призвести до забутих кінцевих точок, які все ще доступні, та збільшити поверхню атаки. Неправильний контроль версій та розгортання може призвести до того, що відставні версії API працюють поряд із поточними версіями. Невиправлені старіші версії спрощують цілі та можуть надавати доступ до виробничих даних.

Недостатня реєстрація та моніторинг. Відстеження всього доступу до кінцевих точок API у потенційно тисячах веб-ресурсів - це непросте завдання. Якщо доступ до API та їх базової інфраструктури недостатньо контролюється та реєструється, зловмисники можуть приховати свою діяльність для виконання багатоетапних атак або навіть підтримувати постійну присутність, не помічаючи цього. Це може призвести до порушення даних, які виявляються та усуваються лише через багато місяців, часто, коли виявляються за скаргами користувачів або зовнішнім тестуванням.

2.4 Фактори для оцінки ймовірності

Після того, як потенційний ризик ідентифіковано, потрібно з'ясувати, наскільки він серйозний, першим кроком є оцінка "ймовірності". На найвищому рівні це приблизний показник того, наскільки ймовірно, що ця конкретна вразливість буде виявлена та використана зловмисником. Не потрібно бути надто точним у цій оцінці. Як правило, достатньо визначити, чи є ймовірність низькою, середньою чи високою.

Існує ряд факторів, які можуть допомогти визначити ймовірність. Перший набір факторів пов’язаний із задіяним зловмисником. Мета - оцінити ймовірність успішного нападу з боку групи можливих нападників. Потрібно пам’ятати, що може існувати декілька зловмисників, які можуть

використовувати певну вразливість, тому зазвичай найкраще використовувати найгірший сценарій. Наприклад, інсайдер може бути набагато більш імовірним зловмисником, ніж анонімний аутсайдер, але це залежить від ряду факторів.

Кожен фактор має набір опцій, і кожен варіант має рейтинг вірогідності від 0 до 9, пов'язаний з ним. Ці цифри будуть використані пізніше для оцінки загальної ймовірності.

Фактори зловмисника

Перший набір факторів пов'язаний із задіяним зловмисником. Мета тут - оцінити ймовірність успішного нападу цієї групи нападників. Використовується найгірший сценарій.

Рівень кваліфікації - наскільки технічно кваліфікованою є ця група зловмиссників?

Відсутність технічних навичок (1),

деякі технічні навички (3),

досвідчений користувач комп'ютера (5),

навички роботи в мережі та програмування (6),

навички проникнення безпеки (9),

Мотив - Наскільки мотивована ця група агентів загроз знаходити та використовувати цю вразливість?

Низька або відсутність винагороди (1),

можлива винагорода (4),

висока винагорода (9)

Можливість - Які ресурси та можливості потрібні цій групі зловмисників, щоб знайти та використати цю вразливість?

Повний доступ або необхідні дорогі ресурси (0),

спеціальний доступ або ресурси (4),

певний доступ або ресурси (7),
доступ та ресурси не потрібні (9)

Розмір - Наскільки велика ця група зловмисників?
Розробники (2),
системні адміністратори (2),
користувачі інтрамережі (4),
партнери (5),
автентифіковані користувачі (6),
анонімні користувачі Інтернету (9)

Таблиця 1 – Фактори зловмисника

Ризик	Фактори зловмисника
Порушений дозвіл на рівні об’єкта	Рівень кваліфікації - навички проникнення безпеки (7) Мотив – висока винагорода (9) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Порушена автентифікація користувача	Рівень кваліфікації - навички проникнення безпеки (7) Мотив - висока винагорода (7) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)

Надмірна експозиція даних	Рівень кваліфікації - навички роботи в мережі та програмування (6) Мотив - висока винагорода (9) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Брак ресурсів та обмеження тарифів	Рівень кваліфікації - досвідчений користувач комп'ютера (5) Мотив - можлива винагорода (4) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Порушена авторизація на рівні функції	Рівень кваліфікації - навички роботи в мережі та програмування (6) Мотив - можлива винагорода (4) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Масове призначення	Рівень кваліфікації - навички проникнення безпеки (9) Мотив - можлива винагорода (4) Можливість - певний доступ або ресурси (7) Розмір - анонімні користувачі Інтернету (9)

Неправильна конфігурація безпеки	Рівень кваліфікації - навички роботи в мережі та програмування (6) Мотив - Мотив - можлива винагорода (4) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Ін'єкція	Рівень кваліфікації - деякі технічні навички (3) Мотив - Мотив - можлива винагорода (4) Можливість - доступ та ресурси не потрібні (9) Розмір - анонімні користувачі Інтернету (9)
Неправильне управління активами	Рівень кваліфікації - навички роботи в мережі та програмування (6) Мотив - можлива винагорода (4) Можливість - певний доступ або ресурси (7) Розмір - анонімні користувачі Інтернету (9)
Недостатня реєстрація та моніторинг	Рівень кваліфікації - деякі технічні навички (3) Мотив - можлива винагорода (4) Можливість - доступ та ресурси не потрібні (9)

	Розмір - анонімні користувачі Інтернету (9)

2. Фактори вразливості

Наступний набір факторів пов'язаний із вразливістю. Мета тут - оцінити ймовірність виявлення та експлуатації конкретної вразливості.

Простота виявлення - наскільки легко цій групі зловмисників виявити цю вразливість?

Практично неможливо (1),

важко (3),

легко (7),

доступні автоматизовані інструменти (9)

Простота експлуатації - наскільки легко цій групі зловмисників використовувати цю вразливість?

Теоретичні (1),

складно (3),

легко (5),

доступні автоматизовані інструменти (9)

Поінформованість - Наскільки добре відома ця вразливість до цієї групи зловмисників?

Невідомо (1),

приховане (4),

очевидне (6),

загальнодоступне (9)

Виявлення вторгнень - Яка ймовірність виявлення вторгнення?

Активне виявлення в додатку (1),
зареєстроване та переглянуте (3),
зареєстроване без перевірки (8),
не зареєстроване (9)

Таблиця 2 – Фактори вразливості

Ризик	Фактори вразливості
Порушений дозвіл на рівні об’єкта	Простота виявлення - доступні автоматизовані інструменти (9) Простота експлуатації - доступні автоматизовані інструменти (9) Поінформованість - приховане (4) Виявлення вторгнень - зареєстроване та переглянуте (3)
Порушена автентифікація користувача	Простота виявлення - доступні автоматизовані інструменти (9) Простота експлуатації - доступні автоматизовані інструменти (9) Поінформованість - очевидне (6) Виявлення вторгнень - зареєстроване та переглянуте (3)
Надмірна експозиція даних	Простота виявлення - доступні автоматизовані інструменти (9) Простота експлуатації - доступні автоматизовані інструменти (9) Поінформованість - очевидне (6) Виявлення вторгнень – не зареєстроване (9)
Брак ресурсів та обмеження тарифів	Простота виявлення - легко (7)

	Простота експлуатації - доступні автоматизовані інструменти (9) Поінформованість - загальнодоступне (9) Виявлення вторгнень - Активне виявлення в додатку (1)
Порушена авторизація на рівні функції	Простота виявлення - важко (3) Простота експлуатації - складно (3) Поінформованість - приховане (4) Виявлення вторгнень - не зареєстроване (9)
Масове призначення	Простота виявлення - складно (3) Простота експлуатації - складно (3) Поінформованість - приховане (4) Виявлення вторгнень - не зареєстроване (9)
Неправильна конфігурація безпеки	Простота виявлення - доступні автоматизовані інструменти (9) Простота експлуатації - легко (5) Поінформованість - приховане (4), Виявлення вторгнень - не зареєстроване (9)
Ін'єкція	Простота виявлення - доступні автоматизовані інструменти (9) Простота експлуатації - легко (5) Поінформованість - приховане (4) Виявлення вторгнень - зареєстроване без перевірки (8),
Неправильне управління активами	Простота виявлення - важко (3)

	Простота експлуатації - складно (3) Поінформованість - приховане (4), Виявлення вторгнень - не зареєстроване (9)
Недостатня реєстрація та моніторинг	Простота виявлення - важко (3) Простота експлуатації - легко (5) Поінформованість - приховане (4) Виявлення вторгнень - не зареєстроване (9)

2.5 Фактори для оцінки впливу

Розглядаючи наслідки успішної атаки, важливо усвідомити, що є два види наслідків. Перший - це „технічний вплив” на програму, дані, які вона використовує, та функції, які вона забезпечує. Інший - це „вплив бізнесу” на бізнес та компанію, яка експлуатує додаток.

Зрештою, вплив на бізнес є більш важливим. Однак ви можете не мати доступу до всієї інформації, необхідної для з'ясування ділових наслідків успішної експлуатації. У цьому випадку надання якомога більш детальної інформації про технічний ризик дозволить відповідному представнику бізнесу прийняти рішення про бізнес-ризик.

Знову ж таки, кожен фактор має набір варіантів, і кожен варіант має рейтинг впливу від 0 до 9, пов’язаний з ним. Ці цифри ми використаємо пізніше, щоб оцінити загальний вплив.

Фактори технічного впливу

Технічний вплив можна розбити на фактори, що відповідають традиційним сферам безпеки: конфіденційність, цілісність, доступність та підзвітність. Мета полягає в тому, щоб оцінити ступінь впливу на систему, в випадку використання вразливості.

Втрата конфіденційності - скільки даних можна розкрити і наскільки вони чутливі?

Розкрито мінімальні нечутливі дані (2),

розкрито мінімальні критичні дані (6),

розкрито великі нечутливі дані (6),

розкрито великі критичні дані (7),

усі дані розкрито (9)

Втрата цілісності - Скільки даних може бути пошкоджено і наскільки вони пошкоджені?

Мінімальні злегка пошкоджені дані (1),

мінімально серйозно пошкоджені дані (3),

великі злегка пошкоджені дані (5),

великі серйозно пошкоджені дані (7),

всі дані повністю пошкоджені (9)

Втрата доступності - скільки послуг можна втратити і наскільки це важливо?

Мінімальні вторинні послуги перервано (1),

мінімальні первинні послуги перервано (5),

великі вторинні послуги перервано (5),

великі первинні послуги перервано (7),

всі послуги повністю втрачено (9)

Втрата підзвітності - Чи простежуються дії зловмисника для конкретної особи?

Повністю відстежується (1),

можливо, простежується (7),

повністю анонімне (9)

Таблиця 3 – Фактори технічного впливу

Ризик	Фактор технічного впливу
Порушений дозвіл на рівні об’єкта	Втрата конфіденційності - розкрито великі критичні дані (7) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - великі первинні послуги перервано (7) Втрата підзвітності - можливо, простежується (7)
Порушена автентифікація користувача	Втрата конфіденційності - розкрито великі критичні дані (7), Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - великі первинні послуги перервано (7), Втрата підзвітності - можливо, простежується (7),
Надмірна експозиція даних	Втрата конфіденційності - розкрито великі критичні дані (7), Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - великі вторинні послуги перервано (5) Втрата підзвітності - можливо, простежується (7)
Брак ресурсів та обмеження тарифів	Втрата конфіденційності - Розкрито мінімальні нечутливі дані (2),

	Втрата цілісності - Мінімальні злегка пошкоджені дані (1) Втрата доступності - всі послуги повністю втрачено (9) Втрата підзвітності - повністю анонімне (9)
Порушена авторизація на рівні функції	Втрата конфіденційності - розкрито великі критичні дані (7) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - великі первинні послуги перервано (7) Втрата підзвітності - можливо, простежується (7)
Масове призначення	Втрата конфіденційності - усі дані розкрито (9) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - великі первинні послуги перервано (7), Втрата підзвітності - можливо, простежується (7),
Неправильна конфігурація безпеки	Втрата конфіденційності - усі дані розкрито (9) Втрата цілісності - всі дані повністю пошкоджені (9) Втрата доступності - всі послуги повністю втрачено (9)

	Втрата підзвітності - можливо, простежується (7)
Ін'єкція	Втрата конфіденційності - розкрито великі критичні дані (7) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - всі послуги повністю втрачено (9) Втрата підзвітності - повністю анонімне (9)
Неправильне управління активами	Втрата конфіденційності - розкрито великі критичні дані (7) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - всі послуги повністю втрачено (9) Втрата підзвітності - Втрата підзвітності - повністю анонімне (9)
Недостатня реєстрація та моніторинг	Втрата конфіденційності - розкрито великі критичні дані (7) Втрата цілісності - великі серйозно пошкоджені дані (7) Втрата доступності - всі послуги повністю втрачено (9) Втрата підзвітності - повністю анонімне (9)

Фактори впливу на бізнес

Вплив на бізнес випливає з технічного впливу, але вимагає глибокого розуміння того, що важливо для компанії, яка запускає додаток. Загалом, ви повинні прагнути підтримати свої ризики з діловим впливом, особливо якщо ваша аудиторія є виконавчим рівнем. Бізнес-ризик - це те, що виправдовує інвестиції у вирішення проблем безпеки.

Багато компаній мають керівництво з класифікації активів та / або посилання на вплив бізнесу, щоб допомогти формалізувати те, що важливо для їх бізнесу. Ці стандарти можуть допомогти вам зосередитись на тому, що справді важливо для безпеки. Якщо вони недоступні, тоді потрібно поговорити з людьми, які розуміють бізнес, щоб зрозуміти, що важливо.

Наведені нижче фактори є загальними напрямками для багатьох підприємств, але ця сфера є ще більш унікальною для компанії, ніж фактори, пов’язані із зловмисником, вразливістю та технічним впливом, тому будемо вважати, що API використовується відомою компанією із великою кількістю клієнтів.

Фінансова шкода - Яку фінансову шкоду спричинить інцидент із ризиком?

Менше витрат на виправлення вразливості (1),

незначний вплив на річний прибуток (3),

значний вплив на річний прибуток (7),

банкрутство (9)

Збиток репутації - Чи призведе експлойт до шкоди репутації, яка може завдати шкоди бізнесу?

Мінімальний збиток (1),

втрата основних рахунків (4),

втрата прихильності (5),

збиток бренду (9)

Невідповідність - Наскільки гучне порушення?

Незначне порушення (2),

значне порушення (5),

гучне порушення (7)

Порушення конфіденційності - скільки особистої інформації можна розкрити?

Одна особа (3),

сотні людей (5),

тисячі людей (7),

мільйони людей (9)

Таблиця 4 – Фактори впливу на бізнес

Ризик	Фактори впливу на бізнес
Порушений дозвіл на рівні об’єкта	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - втрата прихильності (5) Невідповідність - гучне порушення (7) Порушення конфіденційності - тисячі людей (7)
Порушена автентифікація користувача	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - втрата прихильності (5) Невідповідність - гучне порушення (7) Порушення конфіденційності - тисячі людей (7)

Надмірна експозиція даних	Фінансова шкода - значний вплив на річний прибуток (7) Збиток репутації - збиток бренду (9) Невідповідність - гучне порушення (7) Порушення конфіденційності - мільйони людей (9)
Брак ресурсів та обмеження тарифів	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - Мінімальний збиток (1) Невідповідність - Незначне порушення (2) Порушення конфіденційності – (0)
Порушена авторизація на рівні функції	Фінансова шкода - значний вплив на річний прибуток (7) Збиток репутації - втрата прихильності (5) Невідповідність - значне порушення (5) Порушення конфіденційності - тисячі людей (7)
Масове призначення	Фінансова шкода - значний вплив на річний прибуток (7) Збиток репутації - збиток бренду (9) Невідповідність - значне порушення (5) Порушення конфіденційності - тисячі людей (7)

Неправильна конфігурація безпеки	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - збиток бренду (9) Невідповідність - Порушення конфіденційності
Ін'єкція	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - втрата прихильності (5) Невідповідність - значне порушення (5) Порушення конфіденційності - тисячі людей (7)
Неправильне управління активами	Фінансова шкода - незначний вплив на річний прибуток (3) Збиток репутації - втрата прихильності (5) Невідповідність - значне порушення (5) Порушення конфіденційності- тисячі людей (7)
Недостатня реєстрація та моніторинг	Фінансова шкода - значний вплив на річний прибуток (7) Збиток репутації - збиток бренду (9) Невідповідність - значне порушення (5) Порушення конфіденційності - сотні людей (5)

2.6 Визначення серйозності ризику

На цьому етапі оцінка ймовірності та оцінка впливу складаються для розрахунку загальної тяжкості цього ризику. Це робиться шляхом з'ясування, чи є ймовірність низькою, середньою чи високою, а потім роблять те саме для впливу. Шкала від 0 до 9 розділена на три частини:

Таблиця 5 – Шкала оцінювання ризику

Ймовірність та вплив ризику	
Від 0 до <3	Низький
Від 3 до <6	Середній
Від 6 до 9	Високий

Неформальний метод

У багатьох середовищах немає нічого поганого в тому, щоб переглянути фактори та просто взяти відповіді. Потрібно продумати фактори та визначити ключові фактори, які контролюють результат. Можна виявити, що перше враження було неправильним, розглянувши аспекти ризику, які не були очевидними.

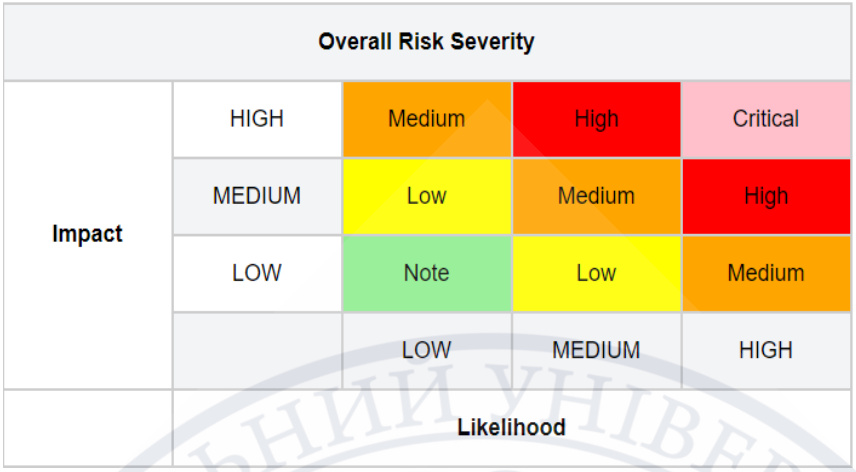
Повторюваний метод

Якщо необхідно робити рейтинги повторюваними, тоді необхідно пройти більш формальний процес оцінки факторів та обчислення результату. В цих оцінках є досить багато невизначеності, і ці фактори мають на меті допомогти тестувальнику досягти розумного результату. Цей процес може підтримуватися автоматизованими інструментами для полегшення обчислення.

Першим кроком є вибір одного з параметрів, пов'язаних з кожним фактором, і введення відповідного номера в таблицю. Тоді просто береться середнє значення балів, щоб розрахувати загальну ймовірність.

Для визначення загальної серйозності ризику використовується така матриця, яка враховує ймовірність та вплив оцінюваного ризику.

Рисунок 2 – Матриця оцінки ризику



Таблиця 6. Оцінювання ймовірності для ризику Порушений дозвіл на рівні об’єкта:

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
7	9	9	9		9	9	4	3
Загальна ймовірність = 7.3 (Висока)								

Таблиця 7. Оцінювання впливу для ризику Порушений дозвіл на рівні об’єкта:

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	7	7		3	5	7	7
Загальний вплив = 6.25 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 8. Оцінювання ймовірності для ризику Порушена автентифікація користувача:

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
7	7	9	9		9	9	6	3
Загальна ймовірність = 7.37 (Висока)								

Таблиця 9. Оцінювання впливу для ризику Порушена автентифікація користувача:

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	7	7		3	5	7	7
Загальний вплив = 6.25 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 10. Оцінювання ймовірності для ризику Надмірна експозиція даних

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
6	9	9	9		9	9	6	9
Загальний вплив = 8.25 (Високий)								

Таблиця 11. Оцінювання впливу для ризику Надмірна експозиція даних

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	5	7		7	9	7	9
Загальний вплив = 7.25 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 12. Оцінювання ймовірності для ризику Брак ресурсів та обмеження тарифів

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
5	4	9	9		7	9	9	1
Загальна ймовірність = 6.63 (Висока)								

Таблиця 13. Оцінювання впливу для ризику Брак ресурсів та обмеження тарифів

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
2	1	9	9		3	1	2	0
Загальний вплив = 3.38 (Середній)								

Загальна серйозність ризику за матрицею – Висока.

Таблиця 14. Оцінювання ймовірності для ризику Порушена авторизація на рівні функції

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
6	4	9	9		3	3	4	9
Загальна ймовірність = 5.88 (Середня)								

Таблиця 15. Оцінювання впливу для ризику Порушена авторизація на рівні функції

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	7	7		7	5	5	7
Загальний вплив = 6.5 (Високий)								

Загальна серйозність ризику за матрицею – Висока.

Таблиця 16. Оцінювання ймовірності для ризику Масове призначення

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
9	4	7	9		3	3	4	9
Загальна ймовірність = 6 (Висока)								

Таблиця 17. Оцінювання впливу для ризику Масове призначення

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
9	7	7	7		7	9	5	7
Загальний вплив = 7.25 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 18. Оцінювання ймовірності для ризику Неправильна конфігурація безпеки

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
6	4	9	9		9	5	5	9
Загальна ймовірність = 7 (Висока)								

Таблиця 19. Оцінювання впливу для ризику Неправильна конфігурація безпеки

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
9	9	9	7		3	9	5	7
Загальний вплив = 6.88 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 20. Оцінювання ймовірності для ризику Ін'єкція

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
3	4	9	9		9	5	4	8
Загальна ймовірність = 6.38 (Висока)								

Таблиця 21. Оцінювання впливу для ризику Ін'єкція

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	9	9		3	5	5	7
Загальний вплив = 6.5 (Високий)								

Загальна серйозність ризику за матрицею – Критична.

Таблиця 22. Оцінювання ймовірності для ризику Неправильне управління активами

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
6	4	7	9		3	3	4	9
Загальна ймовірність = 5.63 (Середня)								

Таблиця 23. Оцінювання впливу для ризику Неправильне управління активами

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	9	9		3	5	5	7
Загальний вплив = 6.5 (Високий)								

Загальна серйозність ризику за матрицею – Висока.

Таблиця 24. Оцінювання ймовірності для ризику Недостатня реєстрація та моніторинг

Фактор зловмисника					Фактор вразливості			
Рівень кваліфікації	Мотив	Можливість	Розмір		Простота виявлення	Простота експлуатації	Поінформованість	Виявлення вторгнень
3	4	9	9		3	5	4	9
Загальна ймовірність = 5.75 (Середня)								

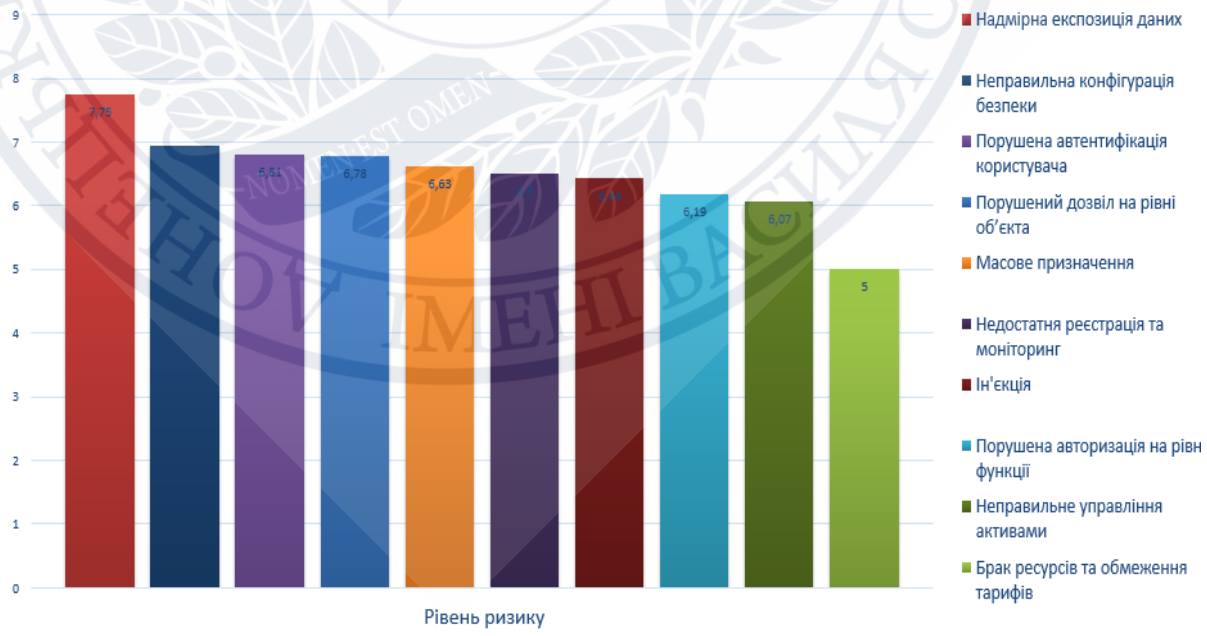
Таблиця 25. Оцінювання впливу для ризику Недостатня реєстрація та моніторинг

Технічний вплив					Бізнес вплив			
Втрата конфіденційності	Втрата цілісності	Втрата доступності	Втрата підзвітності		Фінансова шкода	Шкода репутації	Невідповідність	Порушення конфіденційності
7	7	9	9		7	9	5	5
Загальний вплив = 7.25 (Високий)								

Загальна серйозність ризику за матрицею – Висока.

За результатами даної оцінки, розраховано числовий рівень ризику:

(Загальна ймовірність + Загальний вплив) / 2



2.7 Заходи для зниження ризиків

1. Порушений дозвіл на рівні об’єкта

Весь код, який отримує доступ до об’єктів через API, повинен включати перевірки авторизації. І навпаки, кожного разу, коли ресурси розміщуються у хмарному сховищі, доступному за допомогою API, вам потрібно переконатися, що всі спроби доступу вимагають відповідної авторизації, наприклад за допомогою маркерів доступу. Другий спосіб захисту полягає у використанні непередбачуваних ідентифікаторів, таких як GUID. Коли ми використовуємо послідовні числа в коді, напр. 12345, це означає, що ідентифікатори дуже передбачувані. Зловмисникові не потрібно багато зусиль, щоб перебрати цифри, щоб знайти предмети. GUID складні, їх важко вгадати, і вони не є послідовними. Вони мають такий вигляд: 22345200-abe8-4f60-90c8-0d43c5f6c0f6.

2. Порушена автентифікація користувача

По можливості програми повинні аутентифікувати користувачів за допомогою каталогу, використовуючи id та пароль. Якщо id зберігаються з метою авторизації, вони повинні бути зашифровані. По можливості програми не повинні покладатися на власний механізм автентифікації, а повинні використовувати надані засоби автентифікації.

Паролі повинні мати обмеження, які вимагають мінімального розміру та складності пароля. Складність, як правило, вимагає використання мінімальних комбінацій алфавітних, цифрових та / або небуквено-цифрових символів у паролі користувача (наприклад, принаймні по одному з кожного). Від користувачів слід вимагати періодичної зміни пароля. Користувачам слід заборонити повторно використовувати попередні паролі.

Користувачам слід обмежити певну кількість спроб входу за одиницю часу, а повторні невдалі спроби входу слід реєструвати. Паролі, надані під час невдалих спроб входу, не слід записувати, оскільки це може відкрити пароль користувача тому, хто може отримати доступ до цього журналу. Система не повинна вказувати, чи було ім'я користувача чи пароль неправильним, якщо спроба входу не вдалася. Користувачі повинні бути проінформовані про дату

/ час останнього успішного входу та кількість невдалих спроб доступу до свого облікового запису з того часу.

Слід використовувати єдиний механізм зміни пароля там, де користувачам дозволяється змінювати пароль, незалежно від ситуації. Користувачі завжди повинні вимагати вводити як старий, так і новий пароль при зміні пароля (як і всю інформацію про обліковий запис). Якщо забуті паролі надсилаються користувачам електронною поштою, система повинна вимагати від користувача повторної автентифікації кожного разу, коли користувач змінює свою адресу електронної пошти, інакше зловмисник, який тимчасово має доступ до їх сеансу може просто змінити електронну адресу та попросити надіслати “забутий” пароль.

Якщо додатку потрібно вести списки користувачів для авторизації або перехресних посилань, він повинен зберігати лише (хешоване) ім’я користувача чи мережу кожного користувача. Усі паролі повинні зберігатися у хешованому чи зашифрованому вигляді, щоб захистити їх від впливу, незалежно від того, де вони зберігаються. Хеш-форма є кращою, оскільки вона не оборотна. Паролі ніколи не повинні бути жорстко закодовані в будь-якому вихідному коді. Ключі дешифрування повинні бути надійно захищені, щоб гарантувати, що їх неможливо захопити та використовувати для дешифрування файлу пароля.

Єдиний ефективний прийом безпечної передачі даних - це шифрування всієї транзакції входу за допомогою чогось на зразок SSL. Прості перетворення пароля, такі як хешування його на клієнті перед передачею, забезпечують незначний захист, оскільки хешована версія може бути просто перехоплена та повторно передана, хоча фактичний пароль відкритого тексту може бути невідомий.

В ідеалі весь сеанс користувача повинен бути захищений за допомогою SSL. Якщо це буде зроблено, то ідентифікатор сеансу (наприклад, файли cookie сеансу) не можна вилучити з мережі, що є найбільшим ризиком виявлення для ідентифікатора сеансу. Якщо SSL не є життєздатним з точки

зору продуктивності або з інших причин, тоді ідентифікатори сеансів повинні бути захищені іншими способами. По-перше, їх ніколи не слід включати в URL-адресу, оскільки їх може кешувати браузер, відправляти в заголовку реферала або випадково пересилати «другові». Ідентифікатори сеансів повинні бути довгими, складними, випадковими числами, які неможливо легко вгадати. Ідентифікатори сеансу також можна часто змінювати під час сеансу, щоб зменшити термін дії ідентифікатора сеансу. Ідентифікатори сеансів повинні бути змінені під час переходу на SSL, аутентифікації або інших основних переходів. Ідентифікатори сеансів, вибрані користувачем, ніколи не приймаються.

По можливості, програмам слід уникати зберігання імен облікових записів користувачів або мереж. Системи повинні бути розроблені таким чином, щоб уникнути дозволу користувачам отримати доступ до списку імен облікових записів на сайті. Якщо повинні бути представлені списки користувачів, рекомендується замість цього перерахувати певну форму псевдоніма (псевдонім), що відповідає фактичному акаунту. Таким чином, псевдонім не можна використовувати під час спроби входу в систему або будь-якого іншого злому, який йде після облікового запису користувача. Кешування браузера Дані автентифікації та сеансу ніколи не повинні подаватися як частина GET, замість цього завжди слід використовувати POST. Сторінки автентифікації мають бути позначені всіма різновидами тегу no cache, щоб хтось не використовував кнопку повернення назад у веб-переглядачі користувача для резервного копіювання на сторінку входу та повторне надсилання раніше введених облікових даних. Зараз багато браузерів підтримують прапор autocomplete = false, щоб запобігти збереженню облікових даних у кешах автозаповнення. [13]

3. Надмірна експозиція даних

Кінцеві точки API повинні реалізовувати фільтрацію даних, щоб повертати лише ті властивості, які потрібні для конкретного випадку

використання. Тестування повинно включати перевірку всіх відповідей API, щоб запобігти надмірному впливу. [14]

4. Брак ресурсів та обмеження тарифів

Включайте обмеження розміру та швидкості в специфікації та реалізації API, встановлюйте обмеження ресурсів для середовищ виконання та завжди перевіряйте запити та параметри запиту. [15]

5. Порушена авторизація на рівні функції

Використовуйте послідовний і чітко визначений механізм авторизації, який охоплює всі відповідні кінцеві точки та функції. За замовчуванням забороняють доступ усім і вимагають явних дозволів для ролей та функцій.

6. Масове призначення

Явно вказані білі властивості, до яких клієнт може отримати доступ. Намагайтеся уникати безпосереднього відображення входів клієнта на внутрішні змінні.

7. Неправильна конфігурація безпеки

Визначте та використовуйте повторюваний процес для розгортання належним чином загартованих середовищ. Уникайте надмірно дозвільного спільного використання ресурсів (CORS). Застосовуйте та перевіряйте безпеку у всьому стеку API, приділяючи особливу увагу дозволам на хмарне сховище.

8. Ін'єкція

Фільтруйте (за допомогою білого списку), перевіряйте та кодуйте всі вхідні дані із запитів API, як клієнтські, так і зовнішні системи. Використовуйте параметризовані API для обмеження вхідних даних. Уникайте безпосереднього використання зовнішніх входів у запитах, командах ОС, об’єктно-реляційних відображеннях (ORM) або XML-документах.

9. Неправильне управління активами

Інвентаризація та документування всіх API та хостів API. Забезпечте чітке розділення між виробництвом та іншими середовищами. Створюйте документацію автоматично, щоб вона завжди була актуальною та повною.

Уникайте доступу до виробничих джерел даних із невиробничих середовищ. [16]

10. Недостатня реєстрація та моніторинг

Ведіть докладні та безпечні журнали всіх спроб доступу. Використовуйте систему моніторингу для постійного моніторингу журналів та всього стека API та інфраструктури. Налаштуйте попередження, щоб забезпечити своєчасне реагування на випадки підозрілої діяльності. [17]

ВИСНОВКИ

1. Було досліджено ключові фактори технологій API з точки зору інформаційної безпеки, механізми за якими відбувається передача даних між сервісом та користувачем, а також основні особливості забезпечення безпеки API, які застосовуються розробниками.

2. Використовуючи методологію OWASP Risk Rating Methodology було складено перелік із 10 найпоширеніших ризиків API та оцінено їх. За результатами оцінки з'ясовано, що усі із десяти оцінених ризиків мають високу або критичну ступінь серйозності. В деяких ризиків один із параметрів ймовірності або впливу був "середній", але загальна серйозність для усіх ризиків виявилась високою/критичною. Причиною такого результату є те, що ризики для оцінки було обрано із топу найбільш серйозних за версією проекту OWASP, що й підтвердилося в результаті оцінювання.

3. Для кожного із оцінюваних ризиків було запропоновано відповідні заходи захисту, які повинні впроваджуватися в першу чергу. Використовуючи дані рекомендації можна суттєво знизити ризики інформаційної безпеки для API веб сервісів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Stephen Pritchard. APIs: Risks, Potential and Security Solutions. URL: https://www.infosecurity-magazine.com/news-features/apis-risks-potential-solutions/ (дата звернення: 20.04.2021).

2. Akana - API Security: A Guide To Securing Your Digital Chanels. URL: https://www.roguewave.com/sites/rw/files/documents/white-papers/rw-api-security-fnl.pdf (дата звернення: 21.04.2021).

3. Editorial Team. Як забезпечити безпеку REST API. URL: https://webdevblog.ru/kak-obespechit-bezopasnost-rest-api/ (дата звернення: 24.04.2021).

4. Erez Yalon. Why You Need to Think About API Security. URL: https://www.darkreading.com/application-security/why-you-need-to-think-about-api-security/a/d-id/1335861 (дата звернення: 27.04.2021).

5. Apigee. Building and Managing Secure APIs. URL: https://cloud.google.com/files/apigee/apigee-security-ebook.pdf (дата звернення: 27.04.2021).

6. M. Febri Ramadlan. Introduction and implementation OWASP Risk Rating Management. URL: Riskratingmanagement-170615172835.pdf (owasp.org) (дата звернення: 02.05.2021).

7. API Security: A Guide To Securing Your Digital Channels. URL: https://www.roguewave.com/sites/rw/files/documents/white-papers/rw-api-security-fnl.pdf (дата звернення: 06.05.2021).

8. Santosh Kumar Adidawarpu. Broken Authentication And Session Management And Its Prevention Mechanism. URL: https://www.c-sharpcorner.com/article/broken-authentication-and-session-management-and-its-prevention-mechanism/ (дата звернення: 07.05.2021).

9. Michael Isbitski. Salt Security. API3:2019 Excessive Data Exposure URL: https://salt.security/blog/api3-2019-excessive-data-exposure (дата звернення: 07.05.2021).

10. APIsecurity. API4:2019 — Lack of resources and rate limiting. URL: https://apisecurity.io/encyclopedia/content/owasp/api4-lack-of-resources-and-rate-limiting.htm (дата звернення: 08.05.2021).

11. Sagara Gunathunga. API Security: How to avoid Broken Object Level Authorization & Broken Function Level Authorization. URL: API Security: How to avoid Broken Object Level Authorization & Broken Function Level Authorization | by Sagara Gunathunga | Medium (дата звернення: 12.05.2021).

12. Santiago Rosenblatt. OWASP API #7: Security Misconfiguration. URL: OWASP API #7: Security Misconfiguration | by Santiago Rosenblatt | strike.sh | Medium (дата звернення: 13.05.2021).

13. Erez Yalon. API9:2019 Improper Assets Management. URL: API-Security/0xa9-improper-assets-management.md at master · OWASP/API-Security · GitHub (дата звернення: 15.05.2021).

14. Muzaffer Pasha. Top 5 Ways To Protect Against Data Exposure. URL: Top 5 Ways To Protect Against Data Exposure (traceable.ai) (дата звернення: 16.05.2021).

15. Keycdn. What Is Rate Limiting? URL: What Is Rate Limiting? - KeyCDN Support (дата звернення: 18.05.2021).

16. Rakuten Rapid API. API Security — a Detailed Guide. URL: https://blog.api.rakuten.net/api-security/#API_security_vulnerabilities (дата звернення: 19.05.2021).

17. OWASP API Security top 10 2019. URL: https://raw.githubusercontent.com/OWASP/API-Security/master/2019/en/dist/owasp-api-security-top-10.pdf (дата звернення: 20.05.2021).