Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) *Інститут телекомунікаційних систем*
(повна назва)

Кафедра *Телекомунікаційних систем*
(повна назва)

Освітньо-кваліфікаційний рівень *«бакалавр»*
(назва ОКР)

Напрям підготовки *6.050903 «Телекомунікації»*
(код і назва)

Спеціальність 8(*7).05090302 «Телекомунікаційні системи та мережі»*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Л.О.Уривський
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Некрашу Івану Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів забезпечення якості виготовлення програмних додатків

керівник роботи Лісковський Ігор Олегович, к.т.н., доцент _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » квітня 2016 р. № 1290-с

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: стандарти ISO 9126 та ASVS 3.0.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

а) основна частина:

1) ОГЛЯД СТАНДАРТІВ І ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ;
2) ОГЛЯД ПРОГРАМНИХ ПРОДУКТІВ, ЩО ДОЗВОЛЯЮТЬ КОНТРОЛЮВАТИ ЯКІСТЬ
3) МЕТОДИКА ДЛЯ ОЦІНКИ ЯКОСТІ РОБОТИ ВЕБ-СЕРВІСУ;
4) ВИКОРИСТАННЯ APPSCAN ДЛЯ ТЕСТУВАННЯ ВЕБ - ДОДАТКУ;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Плакат №1 Тема роботи, мета, об'єкт та предмет дослідження, завдання дослідження;
Плакат №2
Плакат №3.
Плакат №4
Плакат №5 Висновки по роботі, напрями подальших досліджень

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз отриманого завдання		
2.	Постановка мети дипломної роботи та розробка попереднього змісту		
3.	Формування вступної частини пояснювальної записки		
4.	Формування першого розділу пояснювальної записки		
5.	Формування другого розділу пояснювальної записки		
6.	Формування третього розділу пояснювальної записки		
7.	Формування четвертого розділу пояснювальної записки		
8.	Оформлення дипломного проекту.		

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Текстова частина дипломної роботи: 64 с., 22 рис.., 7 табл., 18 джерел.

Мета даної роботи полягає в аналізі методів забезпечення якості при розробці програмних додатків.

В даній роботі розглядаються методи забезпечення якості їх переваги та недоліки. Проаналізовано спосіб статистичного контролю якості, та можливість його використання для оцінки якості в процесі розробки програмного забезпечення. Проведено розрахунки, що дозволяють зробити висновки щодо доцільності використання даних методів.

СТАНДАРТ ISO 9126, СПЕЦИФІКАЦІЯ ASVS 3.0, IBM RATIONAL APPSCAN, OWASP.

ABSTRACT

The purpose of this work is to analyze the methods of quality assurance in the development of software applications.

In this paper are considered methods of quality assurance, their advantages and disadvantages. The analysis method of statistical quality control, and the possibility of its use for assessing quality in the software development process. The calculations that allow to draw conclusions on the feasibility of using these methods

STANDARD ISO 9126, SPECIFICATION ASVS 3.0, IBM RATIONAL APPSCAN, OWASP.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ…… 7
ВСТУП…… 9
1 ОГЛЯД СТАНДАРТІВ І ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ…….…… 10
1.1 Життєвий цикл програмного продукту…….. 10
1.1.1 Каскадна модель життєвого циклу……..... 10
1.1.2 V-подібна модель життєвого циклу розробки ПЗ…..... 17
1.2 Огляд стандартів……... 22
1.2.1 Стандарт ISO 9126…… 22
1.2.2 Вимоги до безпеки ПЗ відповідно до ASVS 3.0……. 26
2 ОГЛЯД ПРОГРАМНИХ ПРОДУКТІВ, ЩО ДОЗВОЛЯЮТЬ КОНТРОЛЮВАТИ ЯКІСТЬ…….. 37
2.1 Класифікація аналізаторів вихідного коду…….. 37
2.2 Принципи роботи аналізаторів вихідного коду……. 38
2.3 Огляд можливостей пакету IBM Rational AppScan…… 40
3 МЕТОДИКА КОНТРОЛЮ ДЛЯ ОТРИМАННЯ ОЦІНКИ ЯКОСТІ РОБОТИ ВЕБ-СЕРВІСУ..... 47
3.1 Методика для здійснення контролю за коефіцієнтом відмов…… 47
3.1.1 Визначення коефіцієнту відмов……. 47
3.1.2 Побудова плану контроля з заданими властивостями…….. 48
4 ВИКОРИСТАННЯ APPSCAN ДЛЯ ТЕСТУВАННЯ ВЕБ-ДОДАТКУ…… 51
4.1 Налаштування та запуск програми…….. 51
4.2 Аналіз отриманих результатів…….. 55
ЗАГАЛЬНІ ВИСНОВКИ……... 62
ПЕРЕЛІК ПОСИЛАНЬ…….. 63

ПЕРЕЛІК СКОРОЧЕНЬ

SLC	–	Software life cycle –життєвий цикл програмного забезпечення
ПЗ	–	Програмне забезпечення
ISO	–	International Organization for Standardization – міжнародна організація по стандартизації
ASVS	–	Application Security Verification Standard – стандарт верифікації безпеки додатків
API	–	application programming interface – програмний інтерфейс додатку
URL	–	Uniform Resource Locator – єдиний локатор ресурсу
D.o.S	–	Denial of Service – відмова в обслуговувані
OWASP	–	Open Web Application Security Project
XML	–	eXtensible Markup Language – розширена мова розмітки
SASD	–	Static Application Security Testing – статичне тестування безпеки додатку
DASD	–	Dynamic Application Security Testing – динамічне тестування безпеки додатку
GUID	–	Globally Unique Identifier – унікальний ідентифікатор
LDAP	–	Lightweight Directory Access Protocol – полегшений протокол доступу до каталогів
CWE	–	Common Weakness Enumeration – перелік типічних вразливостей
SSH	–	Secure SHell - мережевий протокол рівня застосунків
CSRF	–	Cross Site Request Forgery – атака типу міжсайтової підробки запитів
AQL	–	Acceptable Quality Limit
RQL	–	Rejection Quality Limit
TLS	–	Transport Layer Security – безпека транспортного рівня

SSH – Secure Sockets Layer – криптографічний протокол для мережі Інтернет

JSON – JavaScript Object Notation – текстовий формат обміну даними, заснований на JavaScript

DOM – Document Object Model – об’єктна модель документа

ВСТУП

Однією з найважливіших задач, що постає перед розробниками програмних додатків є забезпечення якості.

Загалом, процес забезпечення якості можна класифікувати як комплекс дій, що мають на меті впевнитись, що на кожному етапі розробки виконуються вимоги до якості, які були поставлені до продукту. Метою цього процесу є попередження виникнення дефектів завдяки використанню кращіх технік та дотриманню всіх процесів під час розробки. Також до процесу забезпечення якості можна віднести пошук дефектів та їх подальше усунення.

З точки зору технологій розробки задачі забезпечення та контролю якості розділені між собою, а також мають різних виконавців.

Контроль якості проводиться інженерами з тестування і має на меті пошук дефектів з допомогою різних технік та підходів. Натомість інженер з забезпечення якості спрямований на випередження появленя дефектів шляхом налагодження взаємодії між членами команди, грамотної постановки задачі, розрахункам ризиків при розробці.

У даній роботі будуть досліджені саме методи забезпечення якості програмних додатків. Будуть розкриті методи статистичного забезпечення якості та використання автоматизованих комплексів для встановлення слабких місць додатків.

1 ОГЛЯД СТАНДАРТІВ І ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Життєвий цикл програмного продукту

1.1.1 Каскадна модель життєвого циклу

На початку 1970-х років спеціалісти вперше заговорили про наступаючу “кризу ПЗ”. Вона була пов'язана з усе більш підсилюючимся впливом ПО на життя людей, внаслідок чого процеси розробки вимагали постійного вдосконалення. Тому була запропонована концепція життєвого циклу розробки ПО (Software life cycle, SLC). Ця модель реалізує послідовність подій, що відбуваються при розробці ПЗ. Визначення циклу SLC так само як і суперечки щодо сенсу його існування, було предметом багатьох розмов і публікацій, пов'язаних з індустрією ПО. Незважаючи на існування різних точок зору, необхідність в документально підтвердженому процесі розробки ПО залишилася настільки ж актуальною . У 1970 році У.У.Ройс (W.W.Royce) провів ідентифікацію декількох стадій в типовому циклі SLC. Саме Ройс і Баррі Боем (Barry Boehm) припустили. що здійснення контролю кожної стадії процесу розробки призведе до поліпшення якості ПЗ, а також до збільшення продуктивності при розробці програм. Наприклад, робота з проектування інтерфейсів програмного модуля може бути відкладена до тих нір. поки не будуть визначені кінцеві вимоги. Завдяки цьому скорочується кількість можливих переробок. В цьому випадку мова йдеться про каскадну модель життєвого циклу, яку зображено на рисунку 1.1. Дії по розробці ПО "протікають" на графіку послідовно, від блоку до блоку.

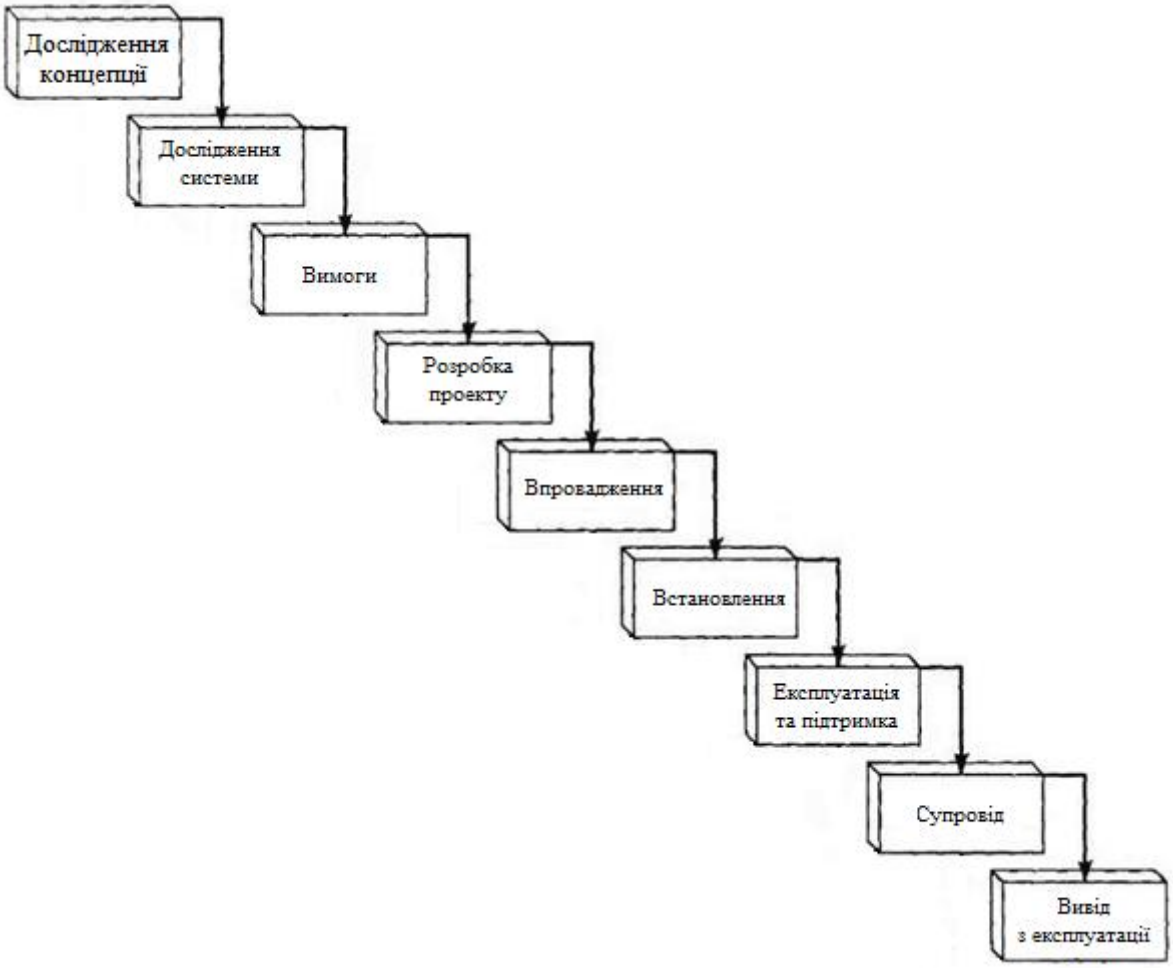

Рисунок 1.1 Каскадна модель розробки ПЗ

Насправді, більшість дій в ході виконання проектів не відбуваються за лінійним законом. Найчастіше розробникам буває необхідно повернутися до попередньої стадії, щоб виконати завдання, які не були в свій час дозволені відповідним чином. Якщо на стадії розробки виявляється відсутність або неправильне формулювання вимог, розробник не просувається вперед, а повертається назад - до стадії специфікацій вимог. Після завершення і корекції специфікацій вимог повторно вводиться і починає виконуватися стадія розробки. Щоб відобразити повторюваний характер розробки ПО, в графіку рисунок 1.1 було додано зворотні стрілки, в результаті чого вийшов графік стандартного життєвого індустріального циклу, показаний на рисунку 1.2.

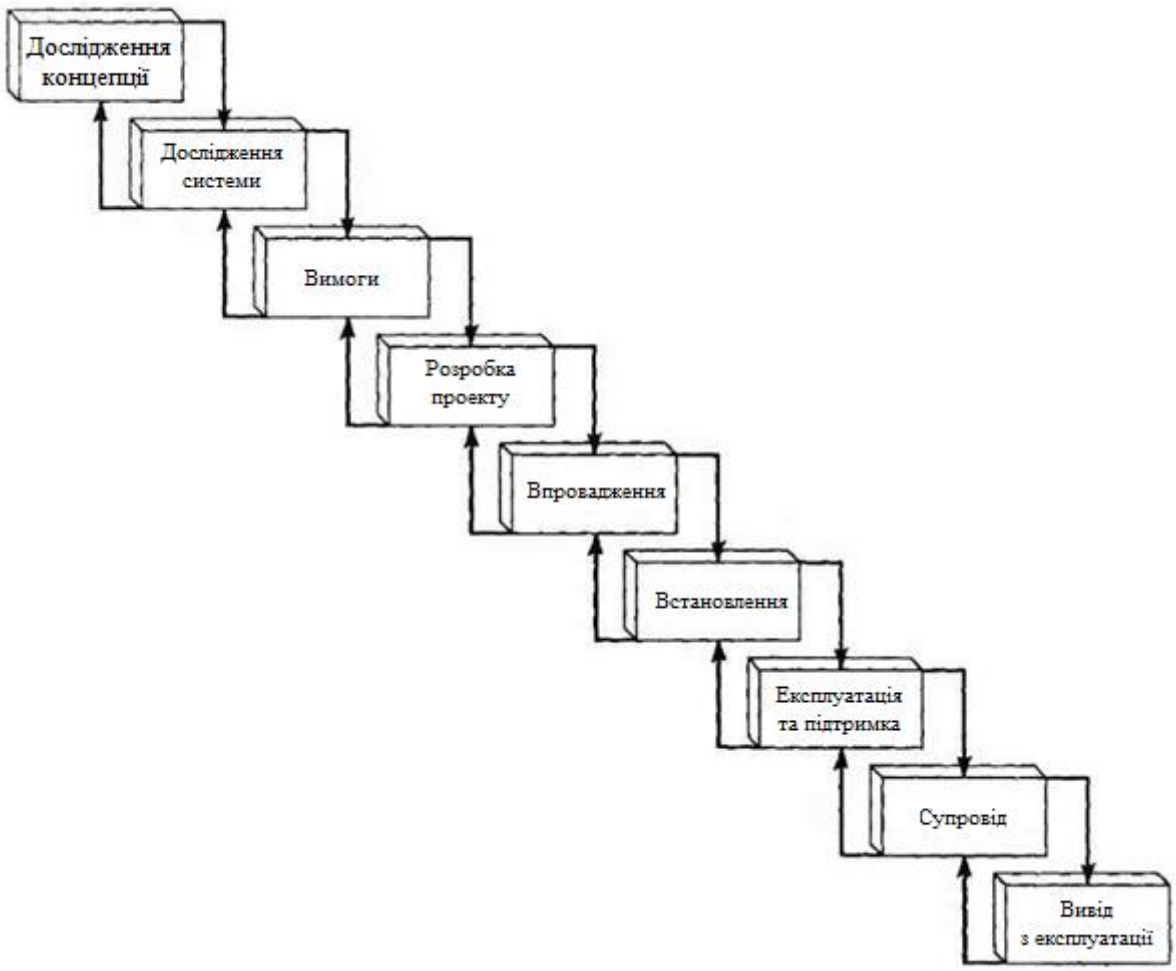

Рисунок 1.2 Повторювана каскадна модель життєвого циклу розробки ПЗ (SLC)

Наведена нижче характеристика являє собою короткий опис кожної фази каскадної моделі (включаючи фази інтеграції):

- дослідження концепції - відбувається дослідження вимог на системному рівні з метою визначення можливості реалізації концепції;
- Процес системного розподілу - може бути пропущений для систем з розробки виключно ПЗ. Для систем, в яких необхідна розробка як апаратного, так і програмного забезпечення, необхідні функції застосовуються до ПЗ і обладнання відповідно до загальної архітектури системи;
- процес визначення вимог - визначаються програмні вимозі для інформаційної предметної області системи, призначення, лінії поведінки, продуктивність і інтерфейси. (В разі необхідності в процес також включено

функціональний розподіл системних вимог до апаратному) 'і програмному)' забезпечення.);

- процес розробки проекту - розробляється і формулнруетсі логічно Послідовна технічна характеристика програмної системи, включаючи структури даних, архітектуру ПЗ, інтерфейсні уявлення і процесуальну (алгоритмічну) деталізацію;
- процес реалізації - в результаті його виконання ескізне опис ПО перетворюється в повноцінний програмний продукт. При цьому створюється вихідний код, база даних і документація, які лежать в основі фізичного перетворення проекту. Якщо програмний продукт являє собою придбаний пакет прикладних програм, основними діями по його реалізації будуть установка і тестування пакету програм. Якщо програмний продукт розробляється на замовлення, основними діями є програмування і код-тестування;
- процес установки-включає установку ПО, його перевірку і офіційну приймання замовником для операційного середовища;
- процес експлуатації та підтримки - має на увазі запуск користувачем системи і поточне забезпечення, включаючи надання технічної допомоги, обговорення питань, що виникли з користувачем, реєстрацію запитів користувача на модернізацію і внесення змін, а також коригування або усунення помилок;
- процес супроводу - пов'язаний з дозволом програмних помилок, несправностей. збоїв, модернізацією та внесенням змін, що генеруються процесом підтримки. Складається з ітерацій розробки і передбачає зворотний зв'язок по про доставлених інформації про аномалії;
- процес виведення з експлуатації - вивід існуючої системи з її активного використання або шляхом припинення її роботи, або завдяки її заміні ноной системою або модернізованою версією існуючої системи;

- інтегральні завдання - включають початок роботи над проектом, моніторинг проекту та його управління, управління якістю, верифікацію і атестацію, управління конфігурацією, розробку документації і професійну підготовку протягом усього життєвого циклу.

Переваги каскадної моделі. Неважко помітити, що каскадна модель має безліч переваг, якщо її використовувати в проекті, для якого вона досить прийнятна. нижче перераховані ці переваги:

- модель добре відома споживачам. які не мають відношення до розробки та експлуатації програм, і кінцевим користувачам (вона часто використовується іншими організаціями для відстеження проектів, не пов'язаних з розробкою ПЗ);
- вона впорядковано впорується зі складнощами і добре спрацьовує для тих проектів, які досить зрозумілі, але все ж важкі для розв'язку;
- вона досить доступна дли розумінні, так як переслідується проста мета - виконати. необхідні дії;
- вона проста і зручна до застосування, так як процес розробки виконується поетапно;
- її структурою може керуватися навіть слабо підготовлений в технічному плані або недосвідчений персонал;
- вона відрізняється стабільністю требоваінй;
- Вона являє собою шаблон, в який можна помістити методи для виконання аналізу, проектування, кодування, тестування і забезпечення;
- Вона добре спрацьовує тоді, коли вимоги до якості домінують над вимогами до витрат і графіку виконання проекту;
- вона сприяє здійсненню суворого контролю менеджменту проекту;
- при правильному використанні моделі дефекти можна виявити на більш ранніх етапах, коли їх усунення ще не вимагає щодо великих витрат;

- вона полегшує роботу менеджеру проекту зі складання плану і комплектації команди розробників;
- вона дозволяє учасникам проекту, що завершив дії на виконуваної ними фазі, взяти участь в реалізації інших проектів;
- Вона визначає процедури з контролю за якістю. Кожні отримані дані піддаються огляду. Така процедура використовується командою розробників для визначення якості системи;
- стадії моделі досить добре визначені і зрозумілі;
- хід виконання проекту легко простежити за допомогою використання тимчасової шкали (або діаграми Гантта), оскільки момент завершення кожної фази використовується в якості стадії.

Недоліки каскадної моделі. Але при використанні каскадної моделі для проекту, який важко назвати відповідним для нас, виявляються такими недоліки:

- в основі моделі лежить послідовна лінійна структура, в результаті чого кожна спроба повернутися на одну або дві фази назад, щоб виправити будь-яку проблему або недолік, призведе до значного збільшення витрат і збою в графіку;
- вона не може запобігти виникненню ітерацій між фазами, які так часто зустрічаються при розробці ПЗ, оскільки сама модель створюється відповідно до стандартного циклу апаратного інжинірингу;
- вона не відображає основну властивість розробки ПО, спрямоване на вирішення задач. Окремі фази строго пов'язані з певними діями, що відрізняється від реальної роботи персоналу або колективів;
- інтеграція всіх отриманих результатів відбувається раптово в завершальній стадії роботи моделі. В результаті такого одиничного проходу через весь процес, пов'язані з інтеграцією проблеми, як правило, дають про себе знати занадто пізно. Отже, виявилися не виявлені раніше

помилки або конструктивні недоліки, підвищити ступінь ризику при невеликому завданню часу на відновлення продукту;

- вона може створити помилкове враження про роботу над проектом.
- у клієнта практично немає можливості ознайомитися з системою заздалегідь, це відбувається лише в самому кінці життєвого циклу. Клієнт не має можливості скористатися доступними проміжними результатами, і відгуки користувачів не можна передати назад розробникам. Оскільки готовий продукт не доступний аж до закінчення процесу, користувач приймає участь в процесі розробки тільки на самому початку - при зборі вимог, і в кінці - під час приймальних випробувань;
- користувачі не можуть переконатися в якості розробленого продукту до закінчення всього процесу розробки. Вони не мають можливості оцінити якість, якщо не можна побачити готовий продукт розробки;
- у користувача немає можливості поступово звикнути до системи. Процес навчання відбувається в кінці життєвого циклу, коли ПО вже запущено в експлуатацію;
- проект можна виконати, застосувавши впорядковану каскадну модель, і привести його у відповідність з письмовими вимогами, що, однак, не гарантує його запуску в експлуатацію;
- кожна фаза є передумовою для виконання наступних дій, що перетворює такий метод в ризикований вибір для систем, які не мають аналогів, так як він не піддасться гнучкому моделюванню;
- для кожної фази створюються результативні дані, які по його завершенню вважаються замороженими. Це означає, що вони не повинні змінюватися на наступних етапах життєвого циклу продукту. Якщо елемент результативних даних будь-якого етапу змінюється (що зустрічається досить часто), на проект зробить негативний вплив зміна графіка, оскільки ні модель, ні план не були розраховані на внесення та дозвіл зміни на більш пізніх етапах життєвого циклу;

- всі вимоги повинні бути відомі на початку життєвого циклу, але клієнти рідко можуть сформулювати всі чітко задані вимоги на цей момент розробки. Модель не розрахована на динамічні зміни в вимогах протягом усього життєвого циклу, так як ці дані "заморожуються". Використання моделі може спричинити за собою значні витрати, якщо вимоги в недостатній мірі відомі або схильна до динамічних змін під час протікання життєвого циклу;
- виникає необхідність в жорсткому управлінні і контролі, оскільки в моделі не передбачена можливість модифікації вимог;
- модель заснована на документації, а значить, кількість документів може бути надмірною;
- весь програмний продукт розробляється за один раз. Немає можливості розбити систему на частини. В результаті взятих розробниками зобов'язань розробити цілу систему за один раз можуть виникнути проблеми з фінансуванням проекту. Відбувається розподіл великих грошових коштів, а сама модель практично не дозволяє повторно розподілити кошти, не зруйнувавши при цьому проект в процесі його виконання;
- відсутня можливість врахувати переробку і ітерації за рамками проекту.

1.1.2 V-подібна модель життєвого циклу розробки ПЗ

V-подібна модель була створена з метою допомогти працюючій над проектом команді в плануванні із забезпеченням подальшої можливості тестування системи. У цій моделі особливе значення надається діям, спрямованим на верифікацію і атестацію продукту. Вона демонструє, що тестування продукту обговорюється, проектується і планується на ранніх етапах життєвого циклу розробки. План випробування приймання замовником розробляється на етапі планування, а компоновочного випробування системи - на фазах аналізу, розробки проекту і т.д. Цей процес розробки планів

випробування позначений пунктирною лінією між прямокутниками V-подібної моделі.

V-подібна модель, показана на рисунку 1.3, була розроблена як різновид каскадної моделі, а значить, успадкувала від неї таку ж послідовну структуру. Кожна наступна фаза починається після завершення отримання результативних даних попередньої фази. Модель демонструє комплексний підхід до визначення фаз процесу розробки ПО. У ній підкреслені взаємозв'язки, що існують між аналітичними фазами і фазами проектування, які передують кодування, після якого йдуть фази тестування. Пунктирні лінії означаают, що ці фази необхідно розглядати паралельно.

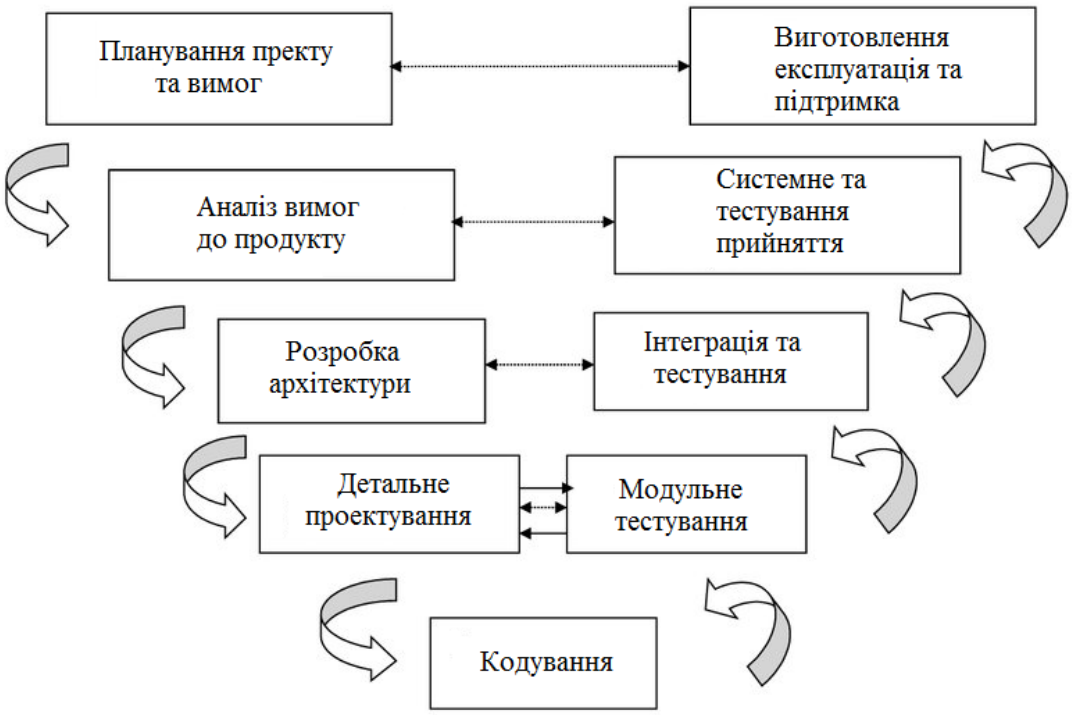

Рисунок 1.3 V-подібна модель життєвого циклу розробки програмного забезпечення

Нижче подано короткий опис кожної фази V-подібної моделі, починаючи від планування проекту та вимог аж до приймальних випробувань:

- планування проекту та вимог - визначаються системні вимоги, а також те, яким чином будуть розподілені ресурси організації з метою їх відповідності

поставленим вимогам. (В разі необхідності на цій фазі виконується визначення функцій для апаратного і програмного забезпечення);

- аналіз вимог до продукту і його специфікації - аналіз існуючої на даний момент проблеми з ПО, завершується повною специфікацією очікуваної зовнішньої лінії поведінки створюваної програмної системи;
- архітектура або проектування на вищому рівні - визначає, яким чином функції ПО повинні застосуватися при реалізації проекту;
- деталізована розробка проекту - визначає і документально обґрунтовує алгоритми для кожного компонента, який був визначений на фазі побудови архітектури. Ці алгоритми надалі будуть перетворені в код;
- розробка програмного коду - виконується перетворення алгоритмів, визначених на етапі деталізований проектування, в готове ПО;
- модульне тестування - виконується перевірка кожного закодуваного модуля на наявність помилок;
- інтеграція і тестування - установка взаємозв'язків між групами раніше поелементно випробуваних модулів з метою підтвердження того, що ці групи працюють так само добре, як і модулі, випробувані незалежно один від одного на етапі поелементного тестування;
- системне і приймальне тестування - виконується перевірка функціонування програмної системи в цілому (повністю інтегрована система), після приміщенні в її апаратне середовище відповідно до специфікацией вимог до ПО.
- виробництво, експлуатація та супровід - ПО запускається у виробництво. На цій фазі передбачені також модернізації та внесення поправок;
- приймальні випробування (на рис. Не показані) - дозволяє користувачеві протестувати функціональні можливості системи на відповідність вихідним вимогам. Після остаточного тестування ПО і довколишній апаратне забезпечення стають робочими. Після цього забезпечується супровід системи.

Переваги V-подібної моделі. При використанні V-подібної моделі для розробки проекту, для якого вона в достатній мірі підходить, забезпечується кілька переваг:

- в моделі особливе значення надається плануванню, направленому на верифікацію і атестацію продукту, що розробляється на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації;
- в моделі передбачені атестація і верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту;
- в V-подібній моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО - перед розробкою компонентів;
- модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню;
- завдяки моделі менеджери проекту може відстежувати хід процесу розробки. Так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою;
- модель проста у використанні (щодо проекту, для якого вона є прийнятною).

Недоліки V-подібної моделі. При використанні V-подібної моделі в роботі над проектом, для якого вона не є в достатній мірі прийнятною, стають очевидними її недоліки:

- з її допомогою непросто впоратися з паралельними подіями;
- в ній не враховані ітерації між фазами;

- в моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу;
- тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести зміни, які не вплинувши при цьому на графік виконання проекту;
- в модель не входять дії, спрямовані на аналіз ризиків.

З метою подолання цих недоліків V-образну модель можна модифікувати, включивши в неї ітераційні цикли, призначені для вирішення змін у вимогах за рамками фази аналізу.

Функції забезпечення якості при використанні вищенаведених моделей зазвичай включають такі види діяльності:

- Планування, що полягає в підготовці плану робіт, в якому знаходять відображення вимоги до якості, розподіляються завдання між виконавцями, складаються графіки і розділяється відповідальність.
- Розробку стратегії, методів і процедур, що складається в створенні нормативних інструкцій, що охоплюють всі етапи проектування, в тому числі, встановлення вимог до системи, програмування, тестування відповідно до потреб конкретного проекту.
- Розвиток засобів, призначених для досягнення високої якості програмного забезпечення, які передбачають адаптацію наявних і створення нових ручних і автоматичних процедур, що дозволяють контролювати його відповідність встановленим вимогам, що стосуються ефективності функціонування і стандартів якості.
- Проведення ревізії, що складаються в аналізі процедур проектування та нормативної документації з точки зору їх відповідності встановленим планам стандартизації процесу розробки програмного забезпечення, в стеженні за виконанням цих планів і наявністю коректує документації.

- Спостереження за випробуваннями програмних засобів, що припускає складання звітів про випробування з аналізом проблем, причин помилок і обгрунтованому коригувальних дій.
- Збереження робочої документації, що стосується звітів про проблеми проектування і програмування системи, контрольних прикладів, протоколів випробувань, тестових даних, оглядів якості та інших аспектів.

1.2 ОГЛЯД СТАНДАРТІВ

1.2.1 Стандарт ISO 9126

Згідно до міжнародного стандарту ISO 9126 якість програмного додатку можна описати, як спроможність програмного продукту при заданих умовах відповідати встановленим та очікуваним вимогам зацікавлених сторін.

При розгляді якості програмного забезпечення з точки зору того ж стандарту ISO 9126 можна виділити поняття внутрішньої якості, яка пов’язана з характеристиками ПЗ самого по собі, без урахування його поведінки, зовнішньої якості, що характеризує ПЗ з точки зору його поведінки, та якості ПЗ при використанні в різних умовах – та якість, яку відчуває користувач за конкретних сценаріїв роботи ПЗ. Для всіх цих поглядів на якість було введено метрики, які дозволяють оцінити його. Також при розробці якісного ПЗ важливою є якість технологічних процесів його розробки. Відношення між цими складовими якості по схемі, прийнятій в ISO 9126, видно на рисунку 1.4.

Стандарт ISO 9126 використовує для опису внутрішньої та зовнішньої якості ПЗ багаторівневу модель. На верхньому рівні виділено 6 основних характеристик якості ПЗ. Кожна характеристика описана за допомогою атрибутів, що до неї входять. У кожного атрибута є набір метрик, що дозволяє оцінити цей атрибут. На рисунку 1.5 наведено набір характеристик та атрибутів, визначених в ISO 9126.

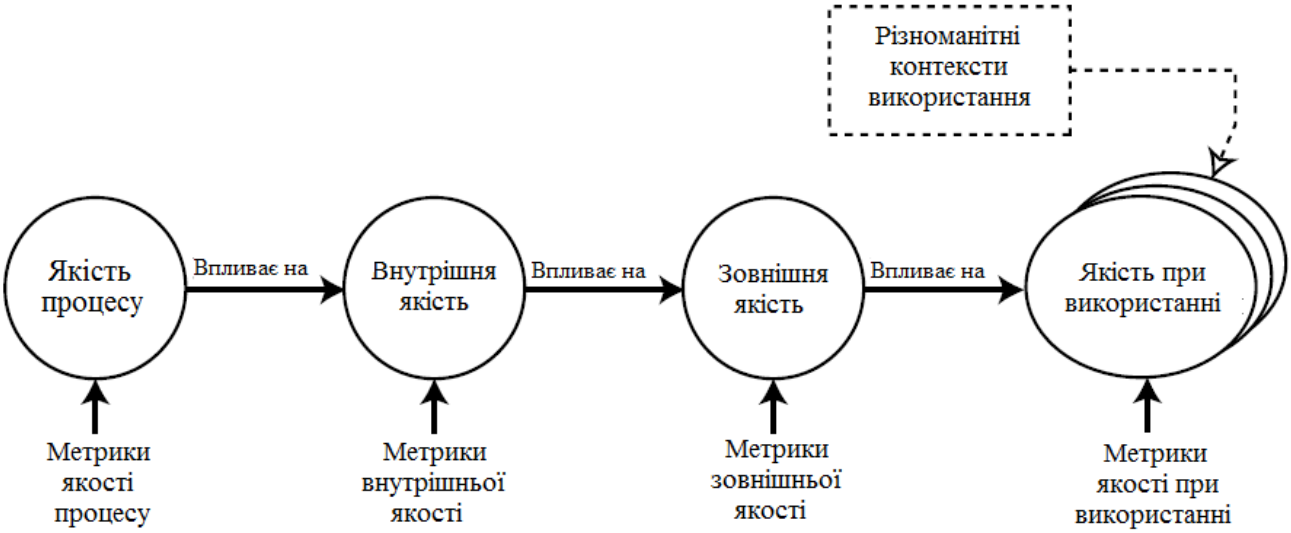

Рисунок 1.4 Основні аспекти якості згідно з ISO 9126.

Рисунок 1.5 Характеристики та атрибути якості ПЗ відповідно до ISO 9126

Нижче приведено визначення цих характеристик та атрибутів по стандарту ISO 9126:2001.

- Функціональність (functionality). Здатність ПЗ в певних умовах вирішувати завдання, потрібні користувачам. Визначає, що саме робить ПЗ, які завдання воно вирішує
  - Функціональна придатність (suitability). Здатність вирішувати потрібний набір завдань.
  - Точність (accuracy). Здатність видавати потрібні результати.\
  - Здатність до взаємодії (interoperability). Здатність взаємодіяти з потрібним набором інших систем.
  - Відповідність стандартам і правилам (compliance). Відповідність ПО наявними індустріальним стандартам, нормативним і законодавчим актам, іншим регулюючим нормам.
  - Захищеність (security). Здатність запобігати неавторизований, тобто без зазначення особи, що намагається його здійснити, і не дозволений доступ до даних і програм.
- Надійність (reliability). Здатність ПЗ підтримувати певну працездатність в заданих умовах.
  - Зрілість, завершеність (maturity). Величина, обернена до частоти відмов ПЗ.
  - Стійкість до відмов (fault tolerance) Здатність підтримувати заданий рівень працездатності при відмовах і порушеннях правил взаємодії з оточенням.
  - Здатність до відновлення (recoverability). Здатність відновлювати певний рівень працездатності і цілісність даних після відмови, необхідні для цього час і ресурси.
  - Відповідність стандартам надійності (reliability compliance). Цей атрибут доданий в 2001 році.
- Зручність використання (usability) або практичність. Здатність ПЗ бути зручним в навчанні і використанні, а також привабливим для користувачів.

- Зрозумілість (understandability). Показник, зворотний до зусиль, витрачених користувачами, щоб сприйняти набір понять, на яких засновано ПО, і їх застосовність для вирішення своїх завдань.
- Зручність навчання (learnability). Показник, зворотний до зусиль, витрачених користувачами щоб навчитися роботі з ПЗ.
- Зручність роботи (operability). Показник, зворотний до зусиль, що вживаються користувачами, щоб вирішувати свої завдання за допомогою програмного забезпечення.
- Відповідність стандартам зручності використання (usability compliance).

- Продуктивність (efficiency) або ефективність. Здатність ПЗ при заданих умовах забезпечувати необхідну працездатність стосовно виділеним для цього ресурсів. Можна визначити її і як відношення одержуваних за допомогою ПЗ результатів до витрачених на це ресурсів.
  - Тимчасова ефективність (time behaviour). Здатність ПЗ видавати очікувані результати, а також забезпечувати передачу необхідного обсягу даних за відведений час.
  - Ефективність використання ресурсів (resource utilisation). Здатність вирішувати потрібні завдання з використанням певних обсягів ресурсів певних видів. Маються на увазі такі ресурси, як оперативна і довгострокова пам'ять, мережеві з'єднання, пристрої введення і виведення, тощо.
  - Відповідність стандартам продуктивності (efficiency compliance).
- Зручність супроводу (maintainability). Зручність проведення всіх видів діяльності, пов'язаних з супровід програм.
  - Зручність проведення аналізу. Зручність проведення аналізу помилок, дефектів і недоліків, а також зручність аналізу на предмет необхідних змін і їх можливих ефектів.
  - Зручність внесення змін (changeability). Показник, зворотний до трудовитрат на проведення необхідних змін.

- Зручність перевірки (testability). Показник, зворотний до трудовитрат на проведення тестування та інших видів перевірки того, що внесені зміни привели до потрібних ефектів.
- Відповідність стандартам зручності супроводу (maintainability compliance).

- Переносимість (portability). Здатність ПЗ зберігати працездатність при перенесенні з одного оточення в інше, включаючи організаційні, апаратні і програмні аспекти оточення.
  - Адаптованість (adaptability). Здатність ПЗ пристосовуватися до різних оточенням без проведення для цього дій, крім заздалегідь передбачених.
  - Зручність установки (installability). Здатність ПЗ бути встановленим або розгорнутим в певному оточенні.
  - Здатність до співіснування (coexistence). Здатність ПЗ співіснувати з іншими програмами в загальному оточенні, ділячи з ним ресурси.
  - Зручність заміни (replaceability) іншого ПО даними. Здатність ПЗ використовуватися замість іншого ПО для вирішення тих же самих завдань в заданому оточенні.
  - Відповідність стандартам переносимості (portability compliance).

1.2.2 Вимоги до безпеки ПЗ відповідно до ASVS 3.0

Згідно до стандарту ASVS 3.0 запропонованого організацією OWASP можна виділити три рівні перевірки безпеки програмних додатків:

- Рівень1 призначений для всіх програм.
- Рівень 2 призначений для додатків, які містять конфіденційні дані, які вимагають захисту.
- Рівень 3 для найбільш важливих додатків - додатків, які містять конфіденційні медичні дані, чутливі дані користувачів або будь-який додаток, який вимагає високого рівня довіри.

Кожен рівень ASVS містить перелік вимог до безпеки. Кожна з цих вимог також може бути співставлена з функціями безпеки конкретних можливостей, які повинні бути вбудовані в програмне забезпечення розробниками.

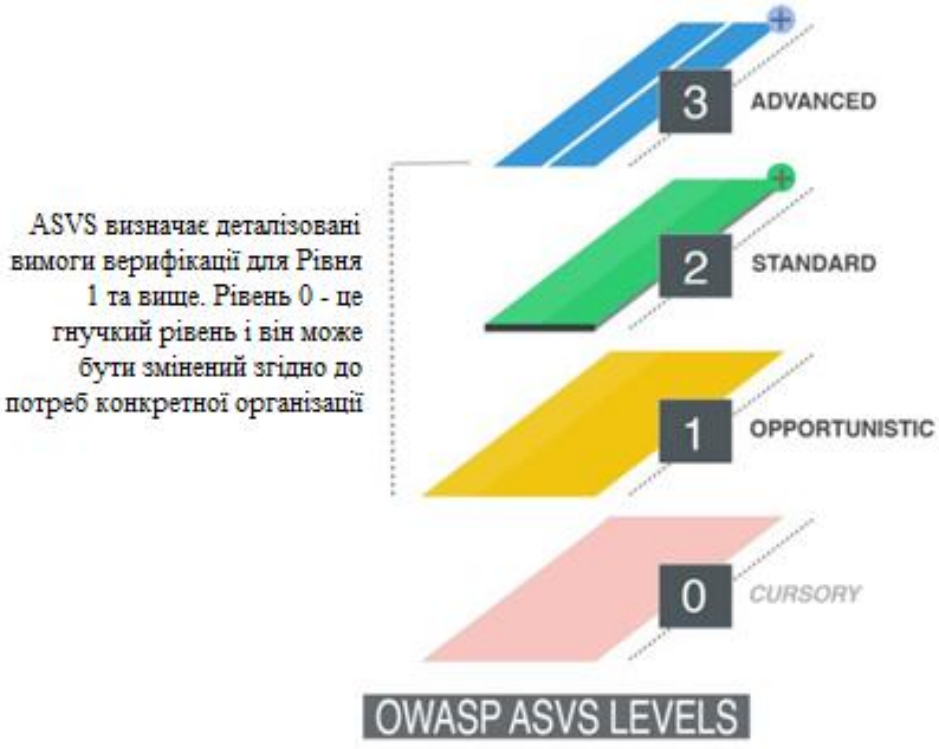

Рисунок 1.6 Рівні OWASP ASVS 3.0

Додаток досягає ASVS 1-го рівня, якщо він адекватно захищає від вразливостей безпеки додатків, які легко виявити, і включені в OWASP Top 10 і інших подібних контрольних списків.

Рівень 1 зазвичай підходить для додатків, де потрібна низька впевненість у правильному використанні засобів контролю безпеки або для швидкого аналізу парку корпоративних додатків або сприяння в розробці пріоритетного списку вимог безпеки в рамках багатофазного зусилля. Елементи управління рівня 1 можуть бути забезпечені або автоматично за допомогою інструментів, або просто вручну без доступу до вихідного коду. Ми розглядаємо рівень 1 як мінімум необхідний для всіх додатків. Загрози для додатка, швидше за все, будуть від зловмисників, які використовують прості і маловитратні методи для виявлення легкодоступних і простих у використанні вразливостей.

Додаток досягає ASVS 2-го рівня, якщо він адекватно захищає від більшості ризиків, пов'язаних з програмним забезпеченням сьогодні. Рівень 2 забезпечує контроль безпеки на місці, ефективний і використовується в додатку. Рівень 2, як правило, підходить для додатків, обробних значні бізнес-бізнес операцій, в тому числі ті, які обробляють інформацію в галузі охорони здоров'я, реалізації бізнес-критичних або чутливих функцій або процесів інших чутливих активів. Загрози застосування 2-го рівня, як правило, будуть кваліфіковані та мотивовані атакуючі зосередивши увагу на конкретних задач з використанням інструментів і методів, які високо практикували і ефективних на виявлення і використання слабких місць в додатках

Рівень 3 забезпечує високий рівень контролю в межах рівнів ASVS. Він, як правило, зарезервований для додатків, що вимагають значних рівнів перевірки безпеки, таких, як ті, що використовуються в таких галузях як: охорона здоров’я, військова, для додатків, що виконують критично важливі функції, де збій може істотно вплинути на діяльність організації, і навіть її живучість. Додаток досягає ASVS 3-го рівня, якщо він адекватно захищає від передових вразливостей безпеки додатків, а також демонструє принципи хорошого дизайну безпеки.

Застосування ASVS 3-го рівня вимагає більш глибокого аналізу, архітектури, кодування і тестування, ніж всі інші рівні.

Далі розглянемо вимоги до безпеки, що висуваються для деяких модулей додатку і визначимо їх для кожного рівня.

Архітектура, проектування та моделювання загроз. Переконайтеся в тому, що перевірений додаток задовольняє наступним вимогам наведеним нижче (таблиця 1.1)

Таблиця 1.1 Вимоги до архітектури додатку

#	Опис	1	2	3
1.1	Переконайтеся, що всі необхідні компоненти програми визначені і відомі.	✓	✓	✓

1.2	Переконайтеся, що всі компоненти, такі як бібліотеки, модулі і зовнішні системи, які не є частиною програми, але на ці додатки покладається додаток є ідентифікованими.		✓	✓
1.3	Переконайтеся в тому, що архітектура високого рівня для додатку була визначена.		✓	✓
1.4	Переконайтеся в тому, що всі компоненти програми визначаються з точки зору бізнес-функцій і/або функцій безпеки, які вони забезпечують.			✓
1.5	Переконайтеся в тому, що всі компоненти, які не є частиною програми, але додаток використовує їх функцій для роботи визначені, і / або функції безпеки, які вони забезпечують є визначеними.			✓
1.6	Переконатися в тому, що додаток захищено від ризиків пов’язаних з підробкою або фальсифікацією, навмисним викриттям інформації, D.o.S атаками або підвищенням привілеїв.			✓
1.7	Перевірте, що всі елементи управління безпекою (в тому числі бібліотеки, які вимагають зовнішніх служб безпеки) мають централізовану реалізацію.			✓
1.8	Переконайтеся, що компоненти відокремлені один від одного за допомогою певного контролю безпеки, такі як мережева сегментація, правила брандмауера або групи безпеки на основі хмарних обчислень.		✓	✓
1.9	Переконайтеся, що програма має чіткий поділ між шаром даних, шаром керування і шаром відображення.		✓	✓
1.10	Переконайтеся в тому, що не існує чутливої бізнес-логіки, секретних ключів або іншої конфіденційної інформації на стороні коду клієнта.		✓	✓

Перевірка вимог до автентифікації. Автентифікація є актом встановлення або підтвердження, що щось (або хтось) має право доступу до певного ресурсу. Переконайтеся в тому, що перевірений додаток задовольняє наступні вимоги високого рівня:

- Перевіряється цифровий код відправника повідомлення.
- Гарантує, що тільки користувачі з певними правами доступу можуть виконувати аутентифікацію і при цьому облікові дані транспортуються в безпечному режимі.

А також нижче наведеним (таблиця 1.2)

Таблиця 1.2 Вимоги до модулю автентифікації

#	Опис	1	2	3
2.1	Всі сторінки і ресурси за замовчуванням вимагають аутентифікації, за винятком тих, які спеціально призначені для публіки (принцип повного посередництва).	✓	✓	✓
2.2	Форми, що містять облікові дані не заповнюються додатком. Попереднє заповнення додатком означає, що облікові дані зберігаються в незашифрованому вигляді, який явно заборонений.	✓	✓	✓
2.3	Всі елементи управління аутентифікацією здійснюються на стороні сервера.	✓	✓	✓
2.4	Переконайтеся в тому, що функціональність заміни пароля включає в себе старий пароль, новий пароль і підтвердження пароля.	✓	✓	✓
2.5	Переконайтеся в тому, що всі рішення аутентифікації можуть реєструватися, без зберігання конфіденційних сеансів ідентифікаторів і паролів. Мають включатися запити з відповідними метаданими, необхідними для досліджень безпеки.		✓	✓

2.6	Переконайтеся, що немає паролів за замовчуванням у використовуваному середовищі розробки програми або будь-яких компонентах, що використовуються додатком (наприклад, «admin / password»).	✓	✓	✓
2.7	Переконайтеся, що всі облікові дані аутентифікації для доступу до послуг зовнішніх по відношенню до додатка зашифровані і зберігаються в захищеному місці.		✓	✓
2.8	Переконайтеся в тому, що блокування облікового запису ділиться на стан м'якого і жорсткого замку, і вони не є взаємовиключними. Якщо обліковий запис тимчасово заблокований м'яко через брут форс атаки, це не повинно скинути стан жорсткої блокування.	✓	✓	✓
2.9	Переконайтеся в тому, що «секретні питання» не порушують закони про конфіденційність і досить сильні, щоб захистити рахунки від зловмисного відновлення.	✓	✓	✓
2.10	Переконайтеся в тому, що система може бути налаштована, щоб заборонити використання заданого кількості попередніх паролів.		✓	✓
2.11	Переконайтеся в тому, що вживаються заходи, щоб заблокувати використання простих паролів і слабких фраз.	✓	✓	✓
2.12	Переконайтеся, що секретні ключі, ключі API і паролі не включені в вихідному коді, або інтернет-сховищах вихідного коду.			✓
2.13	Переконайтеся в тому, що якщо додаток дозволяє користувачам аутентифікуватись, вони можуть проходити перевірку автентичності за допомогою двофакторної перевірки автентичності або більш надійної аутентифікації, або будь-якої подібної схеми,		✓	✓

	яка забезпечує захист імені користувача + пароль розкриття.			
2.14	Переконайтеся в тому, що адміністративні інтерфейси не доступні для ненадійних сторін.	✓	✓	✓
2.15	Переконайтеся в тому, що сеанси анулюються, коли користувач виходить з системи.	✓	✓	✓
2.16	Переконайтеся, що сесії анулюються після певного періоду бездіяльності.	✓	✓	✓
2.17	Переконайтеся, що сеанси анулюються після того, як закінчиться адміністративно-сконфігурований максимальний період часу, незалежно від активності (абсолютний час очікування).	✓	✓	✓

Перевірка вимог щодо управління сесіями. Одним з ключових компонентів будь-якого веб-додатка є механізм, за допомогою якого він контролює і підтримує стан для користувача, що взаємодіє з ним.

Переконайтеся в тому, що перевірений додаток задовольняє наступні вимоги управління сеансами високого рівня:

- Сесії є унікальними для кожної людини і не можуть бути вгадані.
- Сесії анулюються, коли більше не потрібно, або вийшов час.

А також наведеним нижче (таблиця 1.3)

Таблиця 1.3 Вимоги до модуля управління сесіями

#	Опис	1	2	3
3.1	Переконайтеся в тому, що немає користувальницьких менеджерів сеансів, або якщо такий менеджер є, то він стійкий до всіх поширених атак, спрямованих на менеджмент сесій.	✓	✓	✓
3.2	Переконайтеся в тому, що сеанси анулюються, коли користувач виходить з системи.	✓	✓	✓

3.3	Переконайтеся, що сесії анулюються після певного періоду бездіяльності.	✓	✓	✓
3.4	Переконайтеся, що сеанси анулюються після того, як закінчиться адміністративно-сконфігурований максимальний період часу, незалежно від активності (абсолютний час очікування).			✓
3.5	Переконайтеся, що всі сторінки, які вимагають аутентифікації мають простий і видимий доступ до функціональності закінчення сеансу.	✓	✓	✓
3.6	Переконайтеся, що ідентифікатор сеансу ніколи не розкривається в URL, повідомленях про помилки або журналах. Це включає в себе перевірку того, що програма не вміє переписувати URL-адреси, сесійні куки.	✓	✓	✓
3.7	Переконайтеся, що успішна аутентифікація і повторна аутентифікація генерує новий сеанс і новий ідентифікатор сеансу.	✓	✓	✓
3.8	Переконайтеся в тому, що тільки ідентифікатори, які генеруються в рамках програми, визнаються активними цим додатком.		✓	✓
3.9	Переконайтеся, що ідентифікатори досить довгі, випадкові і унікальні в межах сесійної бази.	✓	✓	✓

Обробка шкідливих вхідних даних. Для правильної обробки некоректних вхідних даних додаток має задовольняти нижче наведеним вимогам (таблиця 1.4)

Таблиця 1.4 Вимоги до модуля обробки вхідних даних

#	Опис	1	2	3
4.1	Переконайтеся в тому, що середовище виконання не сприйнятливе до переповнення буфера, або що контроль	✓	✓	✓

	безпеки запобігає переповненню буфера.			
4.2	Переконайтеся в тому, що на стороні сервера введення невірних даних призведе до відхилення запиту та логуванню запиту.	✓	✓	✓
4.3	Переконайтеся, що вхідні перевірки підпрограм виконуються на стороні сервера.	✓	✓	✓
4.4	Переконайтеся, що єдиний орган управління перевірки вхідних даних використовується додатком для кожного типу даних, який є дозволеним.			✓
4.5	Переконайтеся в тому, що всі запити SQL, HQL, OSQL, NoSQL і збережені процедури, виклик цих процедур захищені з використанням підготовлених операторів або параметризації запитів, і, таким чином, не сприйнятливих до SQL ін'єкції	✓	✓	✓
4.6	Переконайтеся в тому, що програма не сприйнятливий до LDAP ін'єкцій, або що контроль безпеки запобігає LDAP ін'єкції.	✓	✓	✓
4.7	Переконайтеся в тому, що програма не сприйнятливий до OS Command Injection, або що контроль безпеки запобігає OS Command Injection.	✓	✓	✓
4.8	Переконайтеся в тому, що програма не вразлива до звичайних атак XML, такі як XPath фальсифікація запитів і XML ін’єкцій.	✓	✓	✓
4.9	Переконайтеся, що програма має захист від атак забруднення параметрів HTTP, особливо якщо фреймворк не розрізняє джерело параметрів запиту (GET, POST, cookies, заголовки, навколишнього середовища і т.д.)		✓	✓
4.10	Переконайтеся, що перевірка на стороні клієнта		✓	✓

	використовується в якості другої лінії захисту, разом з перевіркою на стороні сервера.			
4.11	Переконайтеся в тому, що структуровані дані строго типізовані і звіряються з певною схемою, включаючи допустимі символи, довжини і структури (наприклад, номери кредитних карт або по телефону		✓	✓
4.12	Переконайтеся в тому, що дані, що передаються з одного контексту DOM в інший, використовуют безпечні методи JavaScript, такі як InnerText і value.		✓	✓
4.13	Перевірка при аналізі JSON в браузерах, що JSON.parse використовується для розбору JSON на стороні клієнта. Не використовуйте Eval () для розбору JSON на стороні клієнта.		✓	✓
4.14	Переконайтеся, що дані аутентифікації видаляються зі сховища клієнта, наприклад, DOM в браузері, після того, як сеанс завершується.		✓	✓

Вимоги до криптографії та шифування. При розробці додатку необхідно врахувати наступні вимоги щодо криптографій та шифрування (таблиця 1.5)

Таблиця 1.5 Вимоги до криптографії та шифрування

#	Опис	1	2	3
5.1	Переконайтеся, що всі випадкові числа, випадкові імена файлів, випадковий GUIDs і випадкові рядки генеруються з використанням криптографічних модулів генератора випадкових чисел, коли ці випадкові значення призначені не бути вгаданими зловмисником		✓	✓
5.2	Переконайтеся, що криптографічні алгоритми, використовувані додатком були перевірені згідно до FIPS140-2 або еквівалентного стандарту.	✓	✓	✓

5.3	Переконайтеся в тому, що криптографічні модулі працюють відповідно до опублікованої політики безпеки.			✓
5.4	Переконайтеся, що існує явна політика того, як криптографічні ключі управляються (наприклад, згенеровані, розподілені, анульовані). Переконайтеся в тому, що життєвий цикл ключів дотримуєтться належним чином.		✓	✓
5.5	Персональні дані повинні зберігаються в зашифрованому вигляді. Переконайтеся, що зв'язок йде через захищені канали.		✓	✓
5.6	Переконайтеся в тому, що всі ключі і паролі можуть бути замінені, і генеруються або замінюються під час установки.		✓	✓
5.7	Переконайтеся в тому, що випадкові числа створюються при правильній ентропії, навіть якщо додаток знаходиться під великим навантаженням.			✓

Висновки

У даному розділі було розглянуто каскадну та V-подібну моделі життєвого циклу програмного додатку з їх перевагами та недоліками. Також описано основні вимоги щодо якості розроблюваних додатків відповідно до стандартів ISO 9126 та OWASP ASVS 3.0.

2 ОГЛЯД ПРОГРАМНИХ ПРОДУКТІВ, ЩО ДОЗВОЛЯЮТЬ КОНТРОЛЮВАТИ ЯКІСТЬ

2.1 Класифікація аналізаторів вихідного коду

Аналізатори вихідного коду - клас програмних продуктів, створених для виявлення і запобігання експлуатації програмних помилок у вихідних кодах. Всі продукти, спрямовані на аналіз вихідного коду, можна умовно розділити на три типи:

- Перша група включає в себе аналізатори коду веб-додатків і засоби щодо запобігання експлуатації вразливостей веб-сайтів.
- Друга група - аналізатори вбудованого коду, що дозволяють виявити проблемні місця в початкових кодах модулів, призначених для розширення функціональності корпоративних і виробничих систем.
- Остання група призначена для аналізу вихідного коду на різних мовах програмування, що не відносяться до бізнес-додатків і веб-додатків.

Аналізатори третьої групи призначені для замовників і розробників програмного забезпечення. У тому числі дана група аналізаторів застосовується для використання методології захищеної розробки програмних продуктів. Аналізатори статичного коду знаходять проблеми і потенційно вразливі місця в початкових кодах і видають рекомендації для їх усунення. Варто зазначити, що більшість з аналізаторів відносяться до змішаних типів і виконують функції з аналізу широкого спектра програмних продуктів: веб-додатків, вбудованого коду і звичайного програмного забезпечення. Проте в даному огляді акцент зроблений на застосування аналізаторів замовниками розробки, тому більша увага приділяється аналізатора веб-додатків і вбудованого коду. Аналізатори можуть містити різні механізми аналізу, але найбільш поширеним і універсальним є статичний аналіз вихідного коду - SAST (Static Application Security Testing), також існують методи динамічного аналізу - DAST (Dynamic Application Security Testing), що виконують перевірки коду при його виконанні, і різні гібридні варіанти, що поєднують різні типи аналізів. Динамічний аналіз є

самостійним методом перевірки, який може розширювати можливості статичного аналізу або застосовуватися самостійно в тих випадках, коли доступ до вихідних текстів відсутній. В даному огляді розглядаються тільки статичні аналізатори. Аналізатори вбудованого коду і веб-додатків розрізняються по набору характеристик. У нього входять не тільки якість аналізу і перелік підтримуваних програмних продуктів і мов програмування, а й додаткові механізми: можливість здійснення автоматичного виправлення помилок, наявність функцій щодо запобігання експлуатації помилок без змін коду, можливість поновлення вбудованої бази вразливостей і помилок програмування, наявність сертифікатів відповідності та можливість виконання вимог різних регуляторів.

2.2 Принципи роботи аналізаторів вихідного коду

Загальні принципи роботи схожі для всіх класів аналізаторів: і аналізаторів вихідного коду веб-додатків, і аналізаторів вбудованого коду. Відмінність між цими типами продуктів - тільки в можливості визначити особливості виконання і взаємодії коду із зовнішнім світом, що відбивається в базах вразливостей аналізаторів. Велика частина аналізаторів, представлених на ринку, виконує функції обох класів, однаково добре перевіряючи як вбудовується в бізнес-додатки код, так і код веб-додатків. Вхідними даними для аналізатора вихідного коду є масив вихідних текстів програм і його залежностей (підвантажуваних модулів, використовуваного стороннього програмного забезпечення і т. д.). Як результатів роботи все аналізатори видають звіт про виявлені вразливості і помилки програмування, додатково деякі аналізатори надають функції по автоматичному виправленню помилок. Варто зазначити, що автоматичне виправлення помилок не завжди працює коректно, тому даний функціонал призначений тільки для розробників веб-додатків і вбудованих модулів, замовник продукту повинен спиратися тільки на фінальний звіт аналізатора і використовувати отримані дані для прийняття

рішення по прийманню та впровадження розробленого коду або відправки його на доопрацювання.

Рисунок 2.1. Алгоритм роботи аналізатора вихідних кодів

При проведенні оцінки вихідних текстів аналізатори використовують різні бази даних, що містять опис вразливостей і помилок програмування: Власна база вразливостей і помилок програмування - у кожного розробника аналізаторів вихідних кодів є свої відділи аналітики і досліджень, які готують спеціалізовані бази для аналізу вихідних текстів програм. Якість власної бази - один з ключових критеріїв, що впливає на загальну якість роботи продукту. Крім того, власна база повинна бути динамічною і постійно оновлюваною - нові вектори атак і експлуатації вразливостей, а також зміни в мовах програмування і методи розробки вимагають від розробників аналізаторів виконувати постійні оновлення бази для збереження високої якості перевірки. Продукти зі статичної неоновлюваної базою найчастіше програють в порівняльних тестах. Державні бази помилок програмування - існує ряд державних баз вразливостей, складанням і підтримкою яких займаються регулятори різних країн. Наприклад, в США використовується база CWE - Common Weakness Enumeration, обслуговуванням якої займається організація MITRE, підтримувана в тому числі Міністерством оборони США. Вимоги стандартів і рекомендації по захищеному програмування - існує як ряд державних і галузевих стандартів, що описують вимоги до безпечної розробці свпріложеній, так і ряд рекомендацій і «кращих практик» від світових експертів в області розробки і захисту

програмного забезпечення. Дані документи безпосередньо не описують помилки програмування, на відміну від CWE, але містять перелік методів, які можуть бути перетворені для використання в статичному аналізаторі вихідного коду. Від того, які бази використовуються в аналізаторі, безпосередньо залежить якість проведення аналізу, кількість помилкових спрацьовувань і пропущених помилок. Крім того, аналіз на відповідність вимогам регуляторів дозволяє полегшити і спростити процедуру зовнішнього аудиту інфраструктури та інформаційної системи в тому випадку, якщо вимоги є обов'язковими. Наприклад, вимоги PCI DSS обов'язкові для веб-додатків і вбудованого коду, що працює з платіжною інформацією по банківських картах, при цьому проведення зовнішнього аудиту з виконання PCI DSS здійснюється в тому числі з аналізом застосовуваних програмних продуктів.

2.3 Огляд можливостей пакету IBM Rational AppScan

Основні можливості AppScan:

- Підтримка Flash: завдяки чому він може тестувати додатки побудовані на базі Adobe Flex framework. Протокол AMF також підтримується.
- Тестування прозорого ящика: суть полягає у встановленні спеціального агенту, який допомагає відшукати приховані URL-адреси та інші дефекти.
- Сканування веб-служб: сканування веб-служби є однією з областей, необхідних організаціям для більш ефективної автоматизації.
- Аналізатор безпеки JavaScript: Appscan дозволяє аналізувати html-сторінки на предмет вразливостей, та допомагає користувачеві сфокусуватись на проблемах, зосереджених на клієнтській стороні.
- Звітність: виходячи з ваших вимог, ви можете створювати звіти в бажаних форматах і включати потрібні поля в ньому.
- Настроювані політики сканування: AppScan поставляється з набором певних політик сканування. Ви можете налаштувати політику відповідно до ваших потреб.

- Підтримка інструментів: в наявності є інструменти, такі як Authentication Tester, Token Analyzer і HTTP Request Editor, які можуть бути використані при тестуванні на наявність вразливостей вручну.
- Підтримка Ajax і dojo frameworks.

На рисунку 2.2 зображено алгоритм роботи аналізатора вихідного коду IBM Rational AppScan.

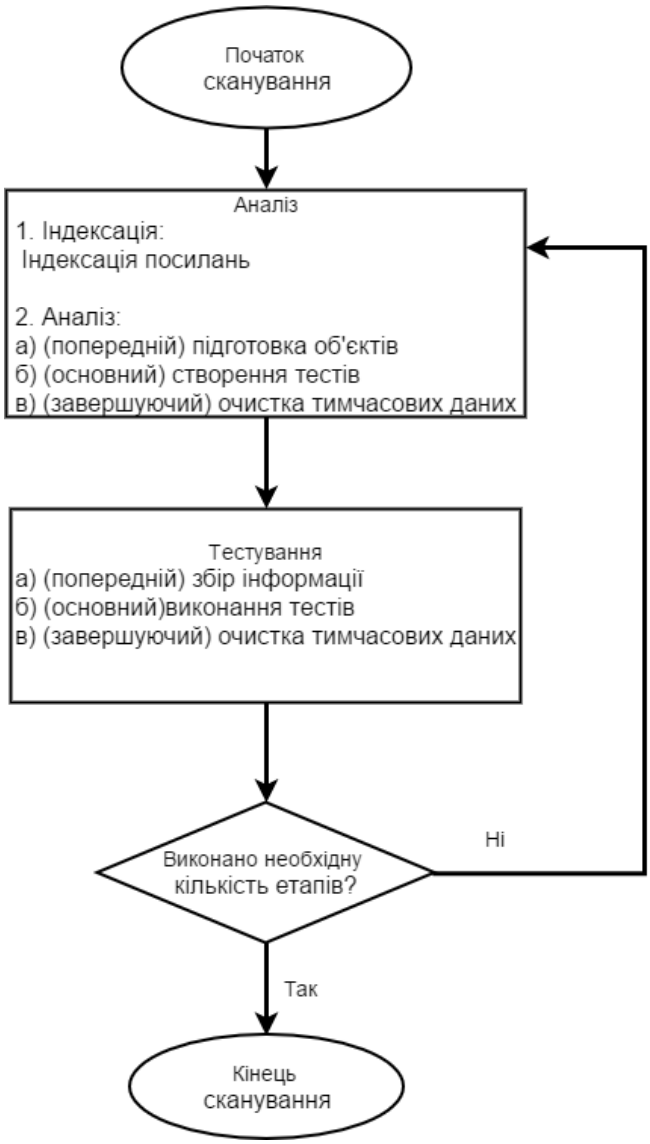

Рисунок 2.2 Алгоритм роботи аналізатора коду

Завдяки обширній базі знань AppScan взмозі відшукати більшість вразливостей перелічених в специфікації OWASP. Вона містить перелік вразливостей, відсортованих за ступенем їх небезпеки (на першому місці найнебезпечніший тип вразливостей). Список не включає в себе всі можливі

типи вразливостей, крім того, часто буває, що ту чи іншу вразливість можна віднести відразу до декількох категорій.

Список OWASP Top 10 є кращою практикою усунення вразливостей Web-додатків, отже розглянемо дані вразливості:

- **Ін’єкція коду.** Ін’єкція коду завжди була однією з найбільш значущих і поширених вразливостей Web-додатків, тому не дивно, що цей тип вразливостей займає верхню строчку списку OWASP. Існує безліч різновидів цієї уразливості, але до сих пір самою сумнозвісною з них є ін’єкція SQL-коду. Вона успішно використовується хакерами протягом більше десяти років. SQL ін'єкція полягає в тому, що зловмисник вводить SQL-команди в поле введення. Якщо код Web-додатка не відфільтровує дані, то на Web-сервері можна запустити SQL-команди і виконувати прямі запити до внутрішньої бази даних в обхід мережевих засобів захисту. Шляхом впровадження SQL-коду зловмисник може дістати таблиці з даними, змінити записи в таблицях і навіть повністю видалити базу даних.
- **Некоректна аутентифікація і управління сеансами.** Друга найбільш небезпечна уразливість в списку OWASP пов'язана з методами аутентифікації і захисту користувальницьких сеансів в Web-додатку. Існує безліч типів цієї уразливості, одним з яких є несанкціоноване використання сеансу. Зловмисник може спробувати вкрасти (несанкціоновано використовувати) Web-сеанс користувача, дізнавшись його секретний ідентифікатор. Знаючи секретний ідентифікатор сеансу, атакуючий може представитися Web-серверу аутентифікованим користувачем і скомпрометувати його обліковий запис. Якщо додаток недостатньо добре захищає ідентифікатори сеансів (наприклад, відображає ідентифікатори всередині URL-адреси замість використання cookie-файлів), то зловмиснику дуже просто отримати ідентифікатор сеансу, обдуривши користувача. Навіть якщо Web-додаток зберігає ідентифікатори сеансів в cookie-файлах, зловмисник все одно може отримати потрібну інформацію з локальних файлів користувача, хитрістю

змусивши його виконати замаскований сценарій. За допомогою простого сценарію можна витягти ідентифікатор сеансу, що зберігається в cookie-файлі на комп'ютері користувача; наприклад, можна отримати ідентифікатор сеансу за допомогою наступного сценарію, введеного в поле пошуку уразливого Web-додатку: <script> alert (document.cookie) </ script>.

- **Міжсайтовий скриптинг.** Міжсайтовий скриптинг (cross-site scripting, XSS) - це ще один різновид атаки на Web-додатки, що зберігає популярність вже багато років. Якщо Web-додаток містить XSS-уразливість, то зловмисник може впровадити на Web-сторінку шкідливий сценарій, що виконується при завантаженні сторінки користувачем. Якщо Web-додаток дозволяє користувачам заходити на Web-сторінку і розміщувати відгуки, то поле для введення коментаря схильне до XSS-атак. Нижче показаний приклад коментаря (This is a great product <script> document.write ('<img scr = http: //evilsite/'+document.cookie>'); </ script >>), що впроваджує в Web-сторінку XSS -сценарій, який намагається дізнатися ідентифікатор сеансу користувача.
- **Небезпечні прямі посилання на об'єкти.** Вразливість, обумовлена наявністю небезпечних прямих посилань на об'єкти, може привести до того, що авторизований користувач Web-додатки може отримати неавторизований доступ до привілейованих функцій і даних. Якщо в коді програми неграмотно або неправильно реалізовані методи роботи з інформаційними об'єктами (наприклад, з файлами, каталогами або ключами баз даних), то користувачі, що не володіють необхідними привілеями, можуть обійти засоби захисту, реалізовані в додатку. Використовуючи цю вразливість, користувачі можуть змінювати значення параметрів таким чином, щоб безпосередньо звертатися до об'єктів, доступ до яких їм заборонено.

- **Небезпечна конфігурація.** Небезпечна конфігурація може бути присутня у всіх компонентах Web-додатки, включаючи платформу (операційну систему), Web-сервер, бази даних або інфраструктуру. Поширеними помилками є запущені без необхідності служби, облікові записи адміністратора з параметрами за замовчуванням, доступне для онлайн-перегляду вміст файлів і директорій. Однак улюбленою темою хакерів є небезпечна обробка повідомлень про помилки. Виведені додатком повідомлення про помилки, дозволяють зловмисникам краще зрозуміти структуру бази даних і підказують, в якому напрямку краще діяти, які команди і сценарії краще намагатися впроваджувати.
- **Витік вразливих даних.** Web-додаток повинен правильно працювати з уразливими даними і захищати їх. Для запобігання витоку даних в процесі їх обробки, передачі і зберігання необхідно забезпечувати захист на всіх рівнях за допомогою надійних процедур контролю доступу, загальновизнаних криптоалгоритмів і методів управління ключами шифрування.
- **Відсутність контролю доступу до функціонального рівня.** Користувачі, що працюють з Web-додатком, можуть мати різні рівні повноважень. У таких випадках базова аутентифікація може не впоратися з тим, щоб не допустити використання користувачами повноважень, функцій і даних, доступ до яких їм заборонено. Якщо захист привілейованих функцій полягає лише в тому, що вони просто заховані в коді програми, то зловмисник, скомпрометувавши обліковий запис з недостатніми привілеями, може спробувати отримати доступ до закритих функціями (і даними), підбираючи виклики до них.
- **Підробка міжсайтових запитів (CSRF).** Підробка міжсайтових запитів (cross-site request forgery, CSRF) заснована на тому, що аутентифікованого користувача Web-додатку обманним шляхом змушують запустити шкідливий сценарій, який виконує дії від імені законного користувача. Наприклад, на комп'ютері користувача може таємно виконуватися CSRF-

сценарій, що відправляє Web-додатком запит на зміну пароля, а після успішної авторизації користувача Web-додаток виконує цей запит. Для захисту від підробки міжсайтових запитів розробники можуть скористатися різними прийомами, включаючи обов'язкове використання в кожному запиті випадкового маркера для перевірки достовірності сеансу або використання випадкових імен полів в різних формах Web-додатку, що робить неможливим створення зловмисником працюючого сценарію.

- **Використання компонентів з відомими уразливостями.** Будь-який сторонній компонент Web-додатку - будь то двійковий або вихідний код, комерційний або Open Source-додаток - повинен перевірятися на відсутність вразливостей. Практично кожен Web-додаток використовує Open Source-компоненти, наприклад, бібліотеку OpenSSL, що забезпечує TLS / SSH-шифрування Web-сайтів (HTTPS). У квітні 2014 року в декількох версіях цієї бібліотеки була виявлена критична вразливість CVE-2014-0160, відома як Heartbleed.
- **Неперевірені перенаправлення і переходи.** Web-додатки часто перенаправляють користувачів на інші сторінки за допомогою певних параметрів. Якщо ці параметри не перевіряються, то зловмисник може обманним шляхом перенаправити користувача на шкідливу Web-сторінку, змусивши його розкрити свій пароль та інші важливі дані. Наприклад, зловмисник може створити повідомлення електронної пошти з фішинговою посиланням, що містить ім'я необхідного Web-сайту. При цьому відповідальний за перенаправлення параметр, що міститься в кінці URL-адреси, може бути не видно користувачеві, в результаті чого користувач з великою ймовірністю клацне за цим посиланням або скопіює її в Web-браузер, так як доменне ім'я в URL-адресі не викличе у нього ніяких підозр.

Висновки

У даному розділі представлено класифікацію аналізаторів вихідного коду, принцип та алгоритми їх роботи. На прикладі IBM Rational AppScan було описано основні можливості аналізаторів, а також типи вразливостей, що можуть бути відшукані завдяки цим програмам.

3 МЕТОДИКА КОНТРОЛЮ ДЛЯ ОТРИМАННЯ ОЦІНКИ ЯКОСТІ РОБОТИ ВЕБ-СЕРВІСУ

3.2 Методика для здійснення контролю за коефіцієнтом відмов

3.1.1 Визначення коефіцієнту відмов

Проведемо розрахунок необхідних параметрів для здійснення контролю за коефіцієнтом відмов сервісу.

Постановка задачі:

1. Визначити тип плану контролю.
2. Побудувати оперативну характеристику для даного плану контролю, якщо основні параметри: $K^i_{відм}$=0,01 (якщо, наприклад, узгоджений ризик оператора 5%, узгоджений ризик користувача 10%)

Для побудови оперативної характеристики проведемо наступне дослідження.

Визначимо, що означає коефіцієнт відмов.

Коефіцієнт відмов, $K^i_{відм}$ –відношення сукупного часу, протягом якого МПДЗК є недоступною, до тривалості періоду випробування або звітного періоду. $K^i_{відм}$ обчислюють окремо для послуг ПД та ДІ за формулою:

$$K^i_{відм} = \frac{T_{недост}}{T_{випр}}, \quad (3.1)$$

де і- індекс виду послуги;

$T_{недост}$ – сукупний час, протягом якого МПДЗК є недоступною;

$T_{випр}$ – тривалість періоду випробування або звітного періоду.

Примітка 1. Зазвичай причиною відмов у з'єднанні з сервером ОПТ є помилки у програмному або апаратному забезпеченні, технічні роботи, що проводяться на сервері, зникнення напруги живлення або інші технічні проблеми. Протягом цих порівняно нетривалих періодів сервер знаходиться у стані неготовності або є недоступним.

Примітка 2. Коефіцієнт відмов визначають шляхом збору статистичних даних окремо для серверів HTTP, FTTP та електронної пошти.

Визначимо в якості генеральної сукупності максимальну кількість запитів до сервісу за 1 добу, яку можемо отримати у оператора на базі статистики за даним параметром, і позначимо її як N. Таким чином, генеральною сукупністю

буде партія розміром N, яка буде перевірена за допомогою вибірки n та кількістю невдалих спроб M.

3.1.2 Побудова плану контролю з заданими властивостями

Найбільш поширений метод побудови планів контролю пов'язаний з заданням рівнів дефектності AQL і RQL і відповідних ризиків постачальника α і користувача β.

Методика заключається в тому, що оперативна характеристика шуканого плану повинна проходити через дві точки (AQL;$1-\alpha$) та (RQL;β), тобто має виконуватися $L(AQL) = 1 - \alpha$ та$L(RQL) = \beta$. Через цілочисельність обох параметрів ni с ці вимоги, як правило, виконуються лише наближено

$$L(AQL) = 1 - \alpha^* \quad (3.2\ а)$$

$$L(RQL) = \beta^* \quad (3.2\ б)$$

де $\alpha^* \approx \alpha$ та $\beta^* \approx \beta$. Для того, щоб задані ризики α і β не були перевищені, потрібне виконання умови $\alpha^* \leq \alpha$ і $\beta^* \leq \beta$, тобто

$$L(AQL) \geq 1 - \alpha \quad (3.3\ а)$$

$$L(RQL) \leq \beta \quad (3.3\ б)$$

Виконання (3.3) означає, що параметри плану контролю nic визначені так, що значення ризиків, тобто дійсного ризику постачальника (оператора) α^* і дійсного ризику користувача (абонента) β^*, в крайньому випадку не перевищують приписаного ризику оператора α та приписаного ризику абонента β.

Для проведення оцінки даного параметра визначимо загальну максимальну кількість запитів до сервісу за день. Дану інформацію отримаємо з аналізу статистики. Позначимо це параметром N.

Побудуємо оперативну характеристику для наступних параметрів (таблиця 3.1).

Таблиця 3.1 Вихідні дані для побудови оперативної характеристики

Параметри	Оперативна характеристика		
	y_1	y_2	y_3
AQL	0,01	0,02	0,01
RQL	0,05	0,05	0,05
A	0,05	0,05	0,1
B	0,03	0,03	0,03
N	15000	15000	15000

Використовуючи методику наведену вище отримаємо наступні оперативні характеристики, що приведені на рисунку 3.1.

Як видно з характеристик найбільшу крутизну має графік зеленого кольору, відповідно даний план контролю буде найбільш зразковим.

Дані оперативні характеристики є оптимальними, в якості критерію оптимальності береться мінімальна кількість проведення випробувань, для забезпечення необхідної точності оцінки. За таким критерієм вартість оцінки якості функціонування мережі буде мінімальною.

Отримані значення піспредставлені у таблиці 3.2.

Таблиця 3.2 Параметри n і c, синтезованих планів контролю

Оперативна характеристика	Параметри	
	n	c
y_1	236	5
y_2	429	13
y_3	167	3

де n-кількість елементів у вибірці (кількість випробувань),
c – приймальне число (максимальна кількість дефектів, при якій можливо прийняти партію).

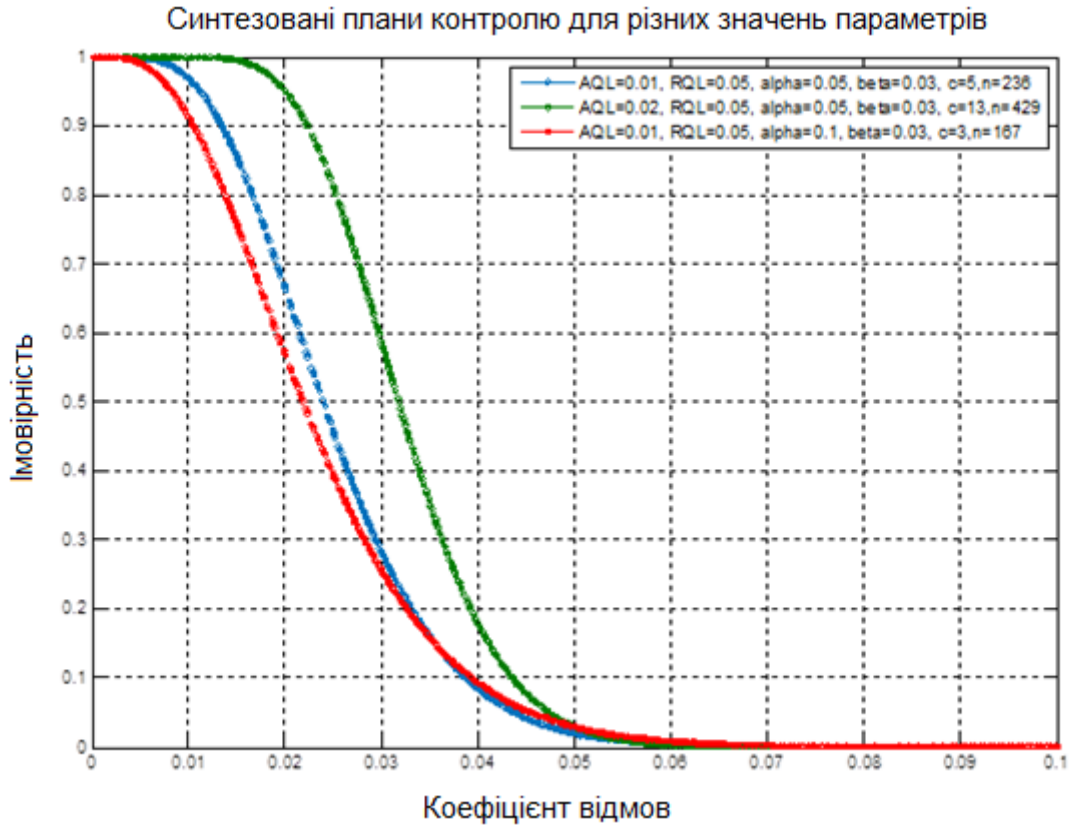

Рисунок 3.1. Синтезовані плани контролю для різних значень параметрів оперативних характеристик

Висновки

В даному розділі було розкрито суть використання статистичних методів для забезпечення якості. Було проведено розрахунок плану вибіркового контролю за альтернативною ознакою. Для даного плану було визначено ризики користувача та постачальника послуги, необхідну кількість випробувань, а також побудовано оперативну характеристику.

4 ВИКОРИСТАННЯ APPSCAN ДЛЯ ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

4.1 Налаштування та запуск програми

Основною задачею програмного комплексу AppScan є проведення тестування в автоматичному режимі веб-додатків або веб-сервісів на наявність в них вразливостей в системі безпеки.

В рамках дослідження на прикладі ресурсу https://demo.testfire.net буде показано всі фази проведення тестування.

Після створення нового проекту, майстер налаштування сканування запропонує ввести URL-адресу веб-додатку, який підлягає тестуванню та при необхідності додаткові налаштування. Після цього буде встановлено зв’язок з сервером, де розташовано веб-додаток (рисунок 4.2).

У разі успішного з’днання AppScan дозволить перейти до наступного кроку, а саме етапу авторизації в додатку, що тестується.

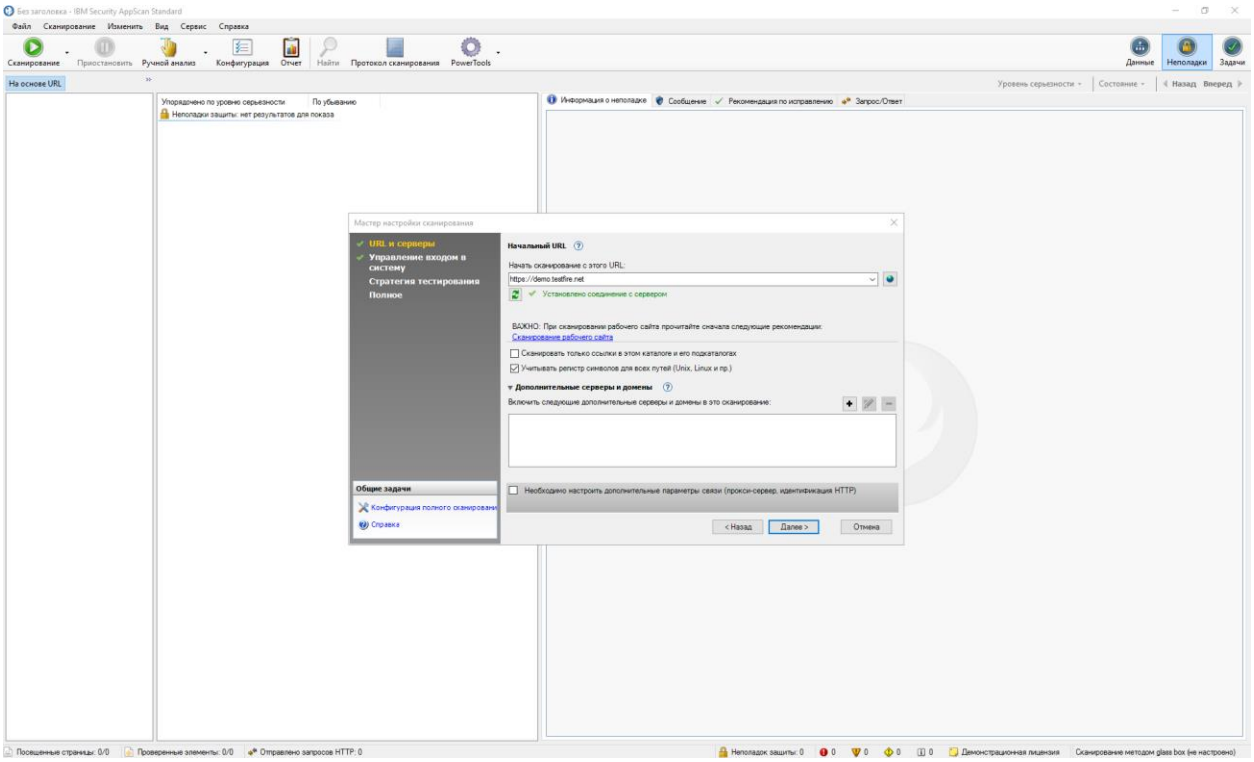

Рисунок 4.1 Загальний вигляд вікна AppScan

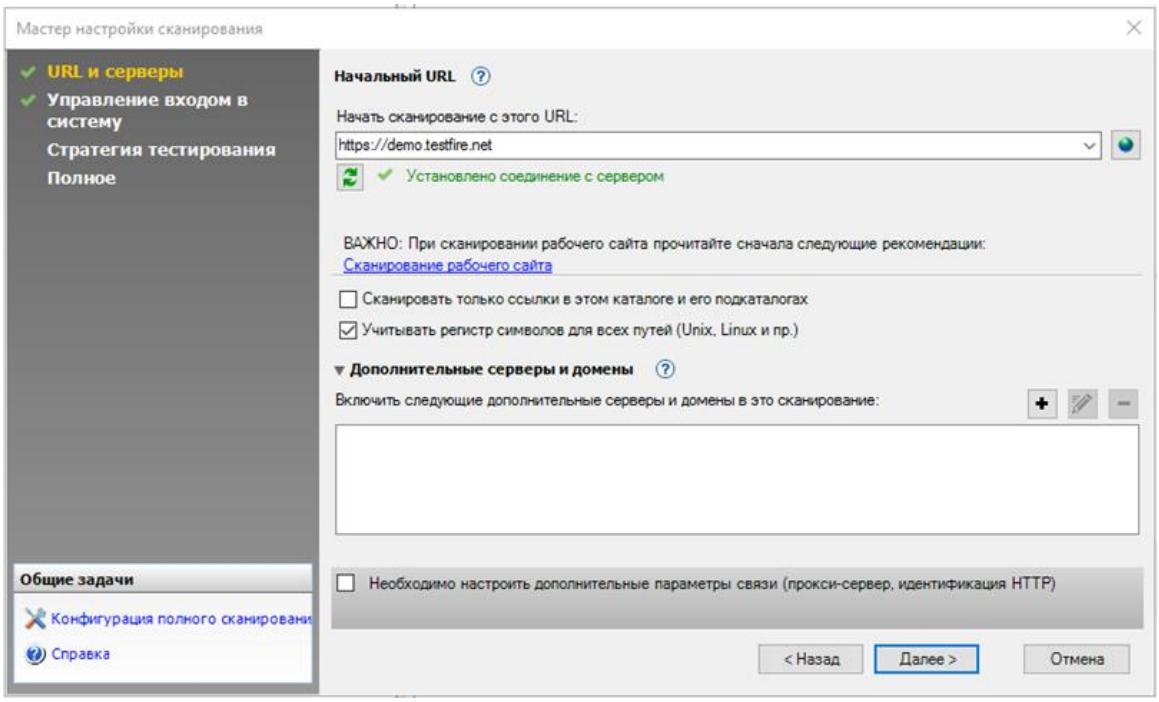

Рисунок 4.2 Майстер налаштування сканування

Далі необхідно пройти процедуру авторизації у веб-додатку. Для цього в комплекті з AppScan э стандартний браузер (рисунок 4.3). У даному вікні треба ввести облікові дані (логін та пароль).

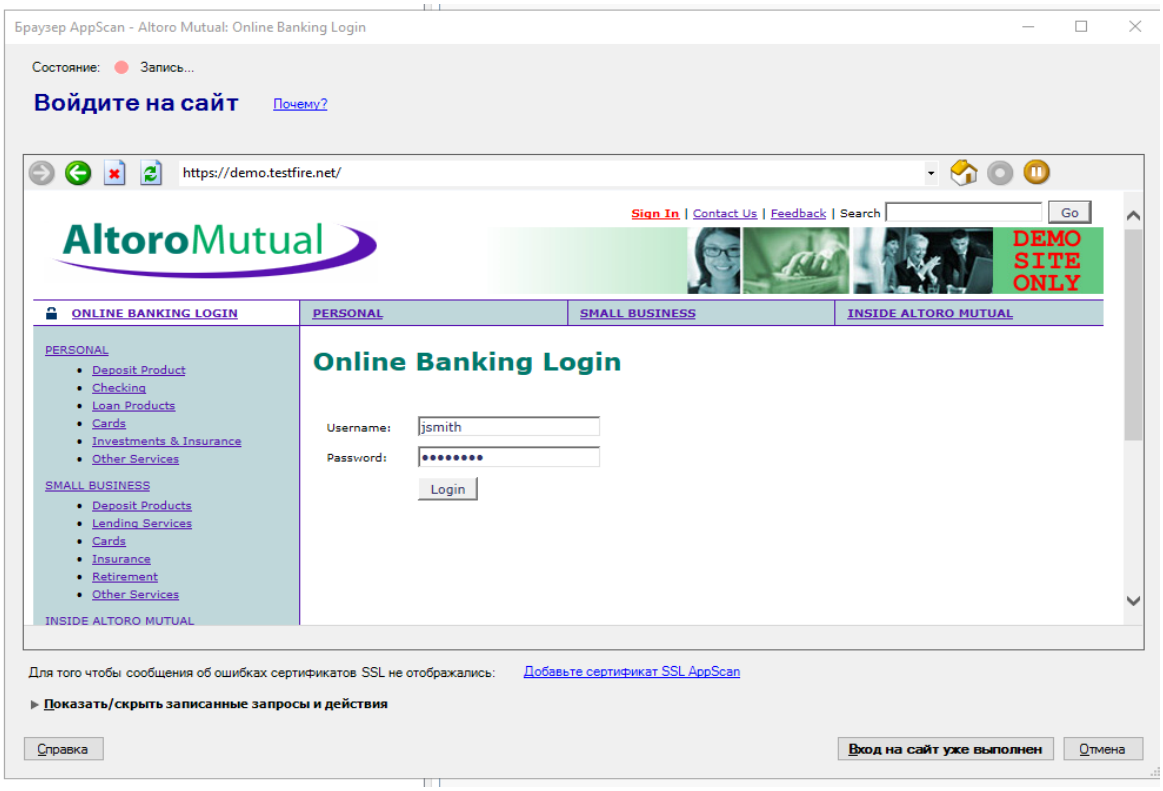

Рисунок 4.3 Авторизація у веб-додатку

В результаті виконаних дій AppScan збереже алгоритм авторизації у вигляді запитів до сервера, які треба виконати. Проведення входу до облікового запису допоможе провести більш детальне сканування додатку.

Після виконання цих дій почнеться сканування веб-додатку, яке відбувається в декілька етапів:

- індексація посилань(створення дерева веб-додатку)
- аналіз, що полягає в підготовці об’єктів та створенні тестів
- тестування, де виконується збір інформації, виконання тестів та очистка тимчасових даних.

На рисунку 4.4 зображено вікно програми по завершенню етапу створення дерева додатку. На рисунку 4.5 відображено, безпосередньо, структура додатку, а на рисунку 4.6 список запитів з параметрами, які було виконано для її побудови.

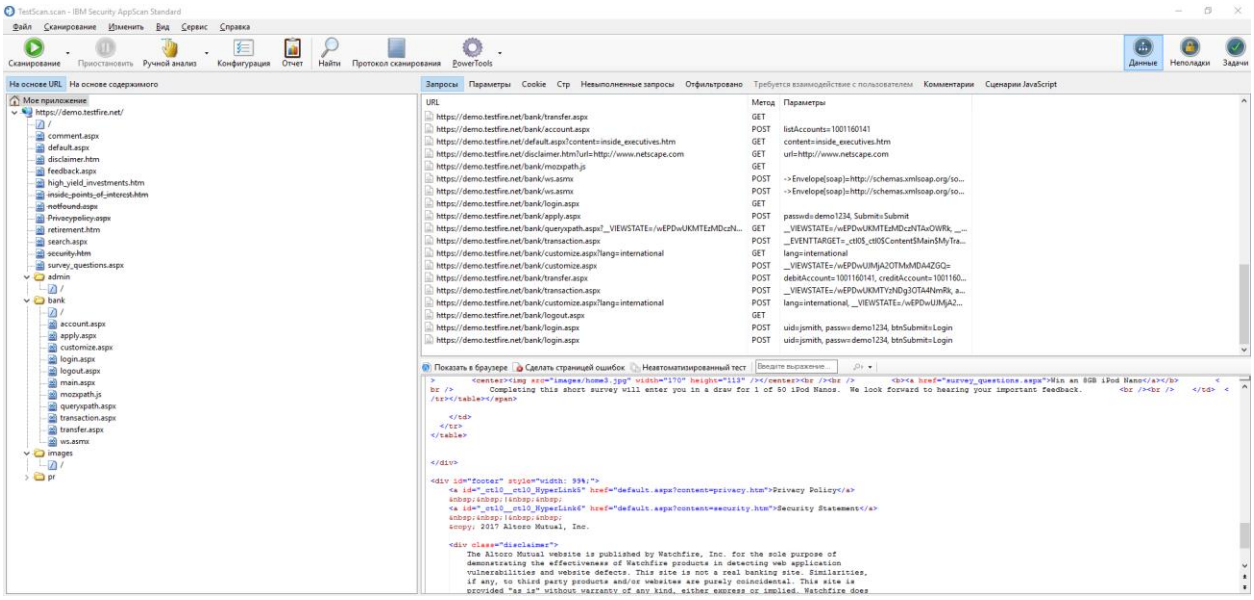

Рисунок 4.4 Вигляд вікна AppScan після аналізу додатку

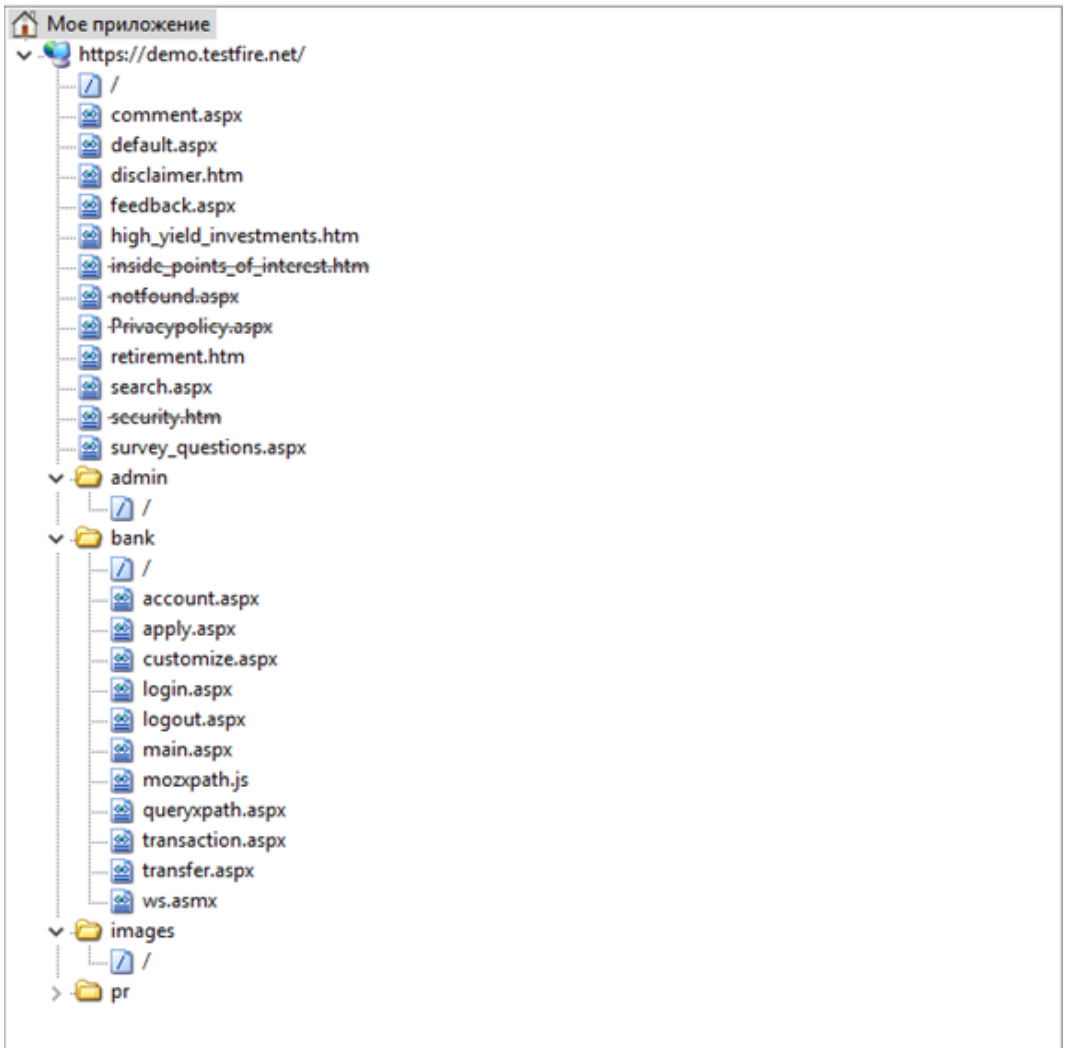

Рисунок 4.5 Дерево сторінок веб-додатку

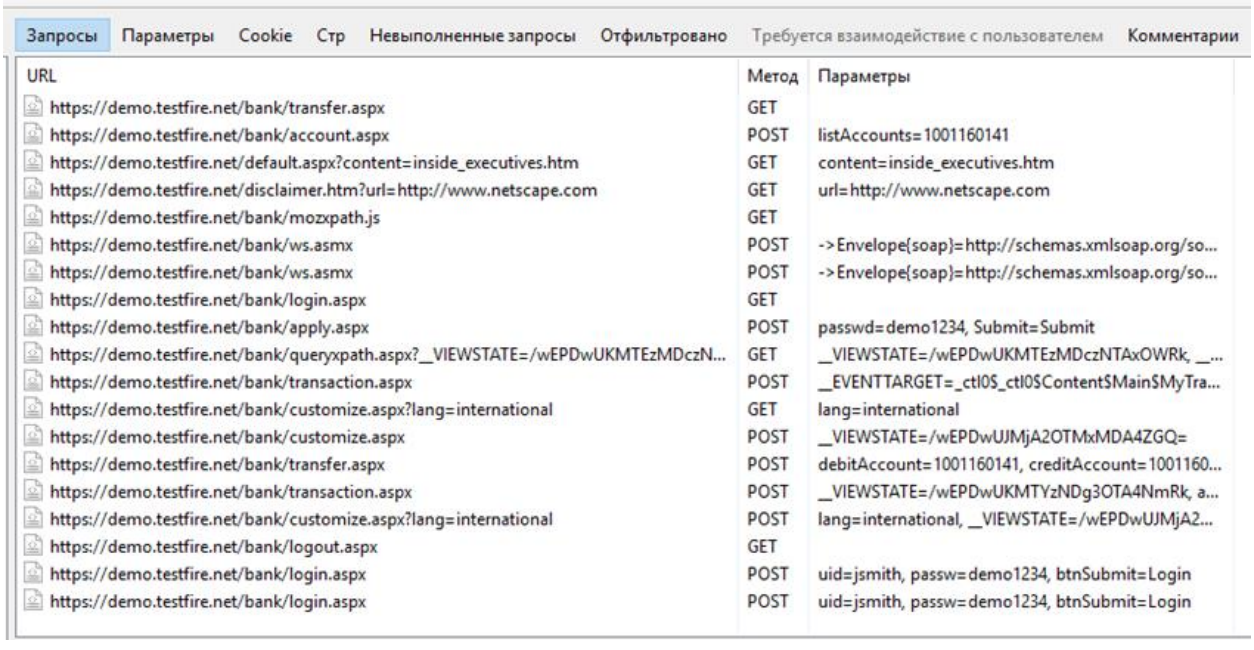

Рисунок 4.6 Список виконаних запитів

4.2 Аналіз отриманих результатів

На рисунку 4.7 приведено вигляд вікна IBM Rational AppScan після завершення процесу сканування.

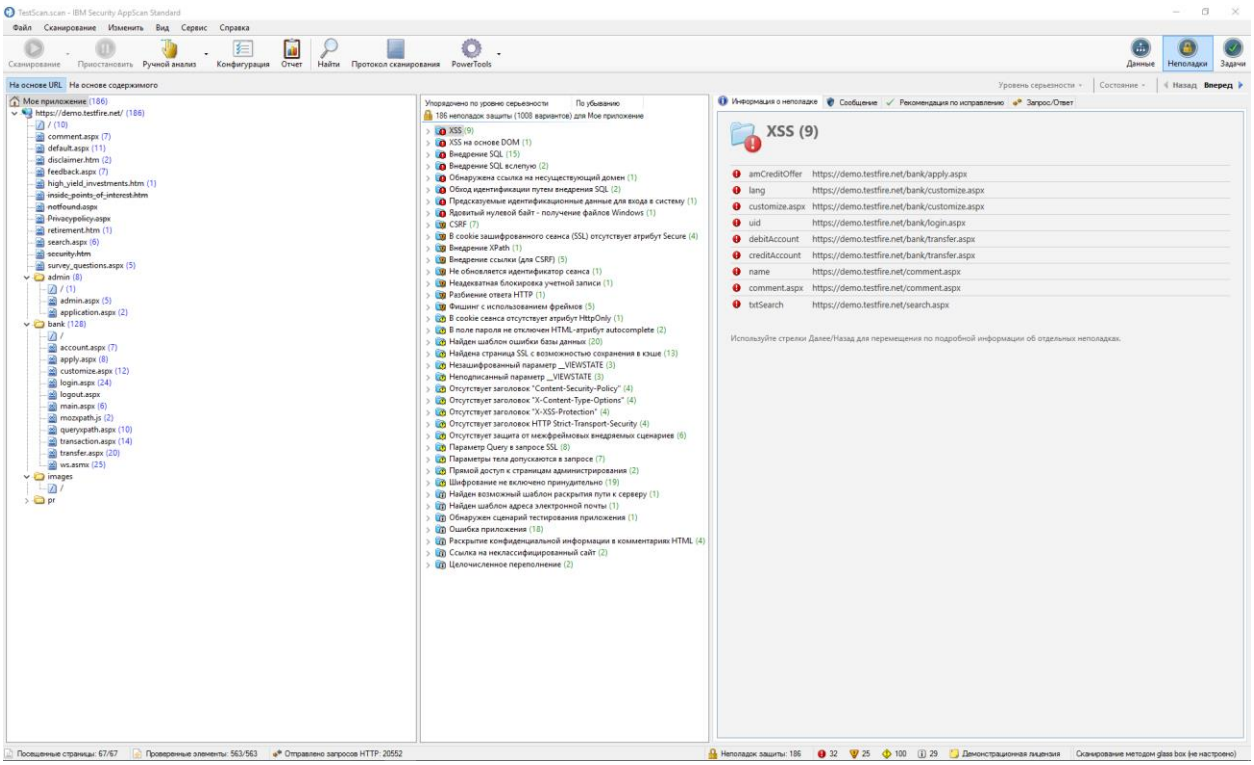

Рисунок 4.7 Вікно AppScan по завершенню перевірки

Дане вікно розділене на три області. В першій відображено структуру веб-додатка,на основі якої проводилося тестування. В другій (рисунок 4.8), безпосередньо, виводиться результат сканування, який має виглядл випадаючого списку з вразливостями, а також їх кількістю. В третій області розкривається більш детальна інформація по кожній відшуканій проблемі.

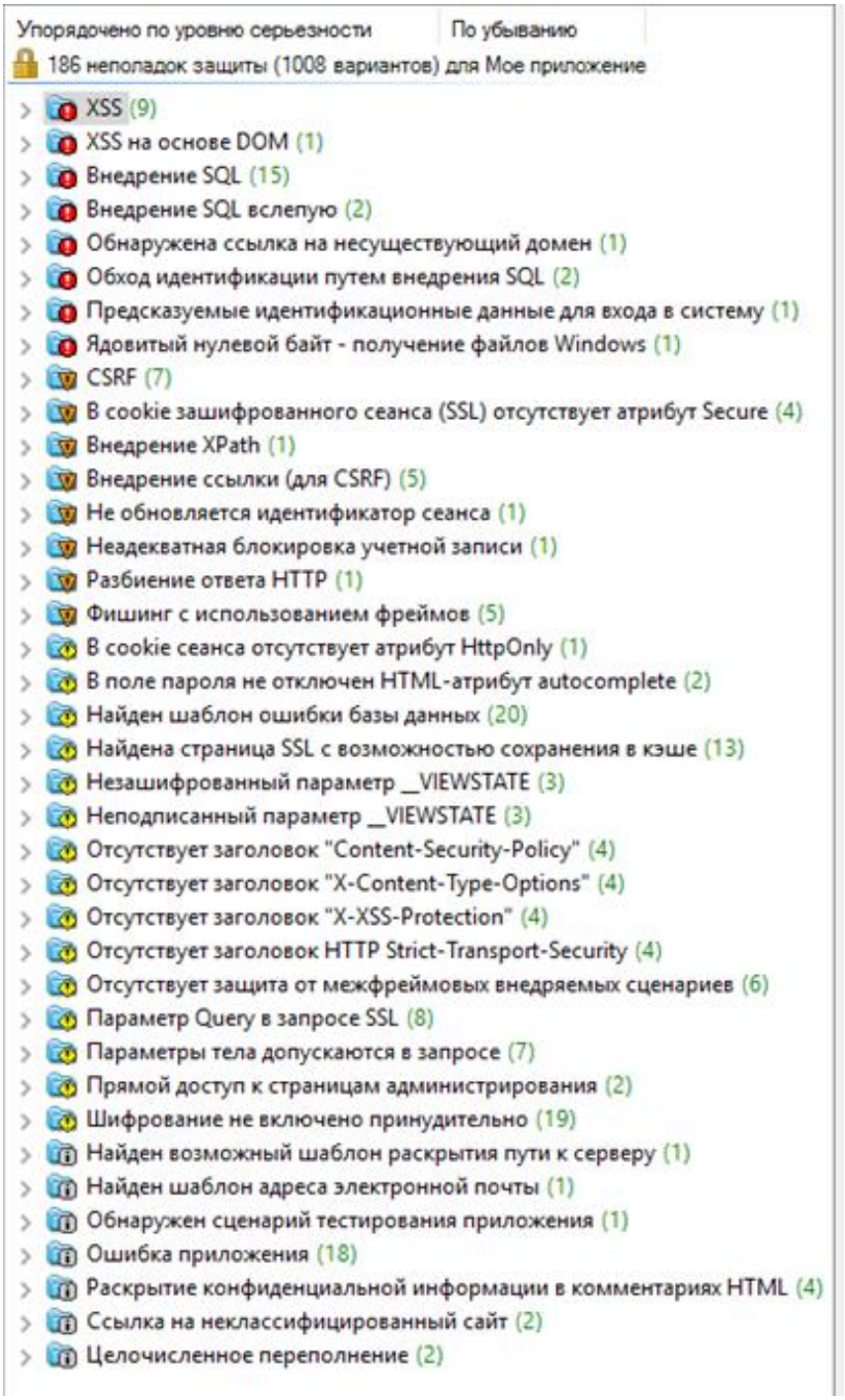

Рисунок 4.8 Списов знайдених вразливостей

Для кожної знайденої вразливості можна подивитись детальну інформацію про те, як було модифіковано запит (рисунок 4.9) і яку відповідь в результаті було отримано (рисунок 4.10):

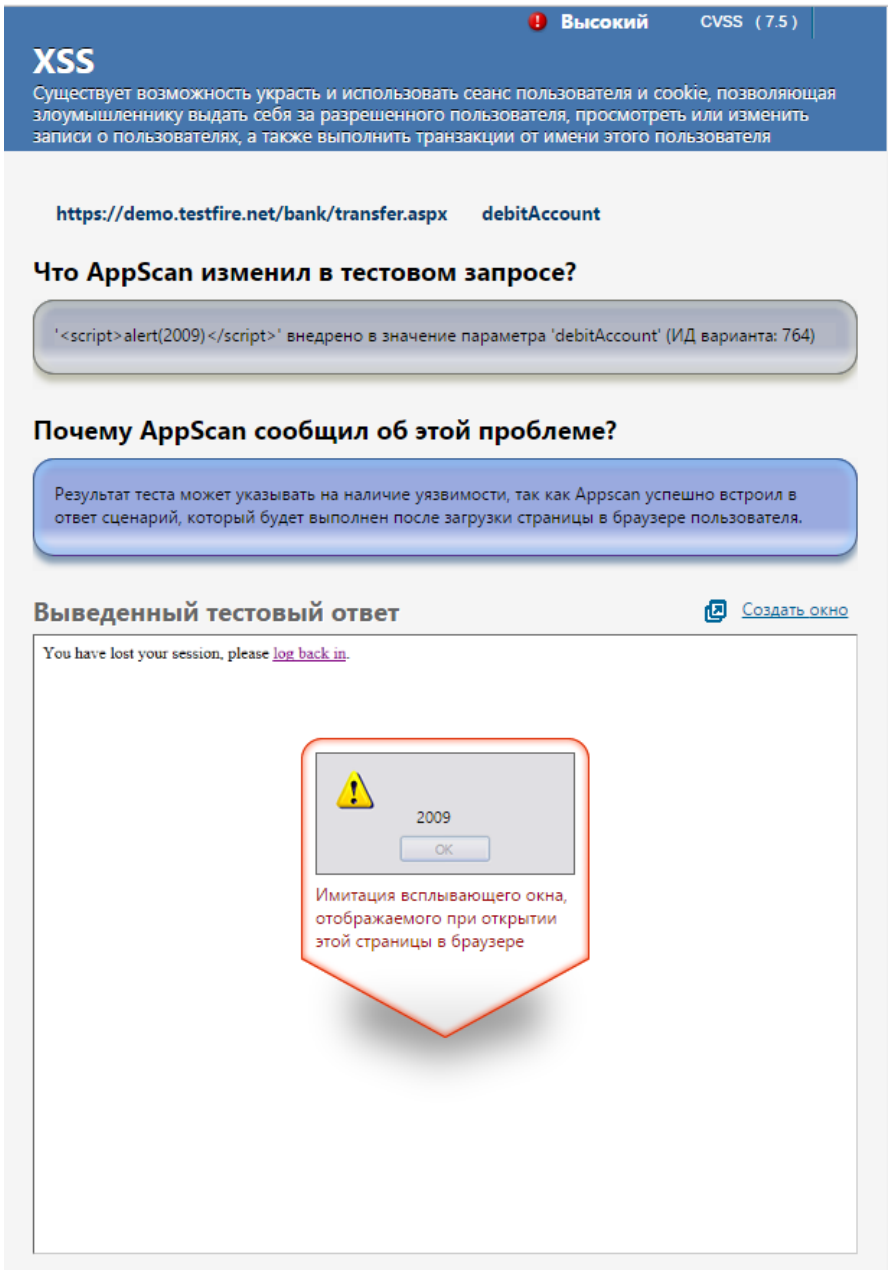

Рисунок 4.9 Змінений запит та відповідь на нього

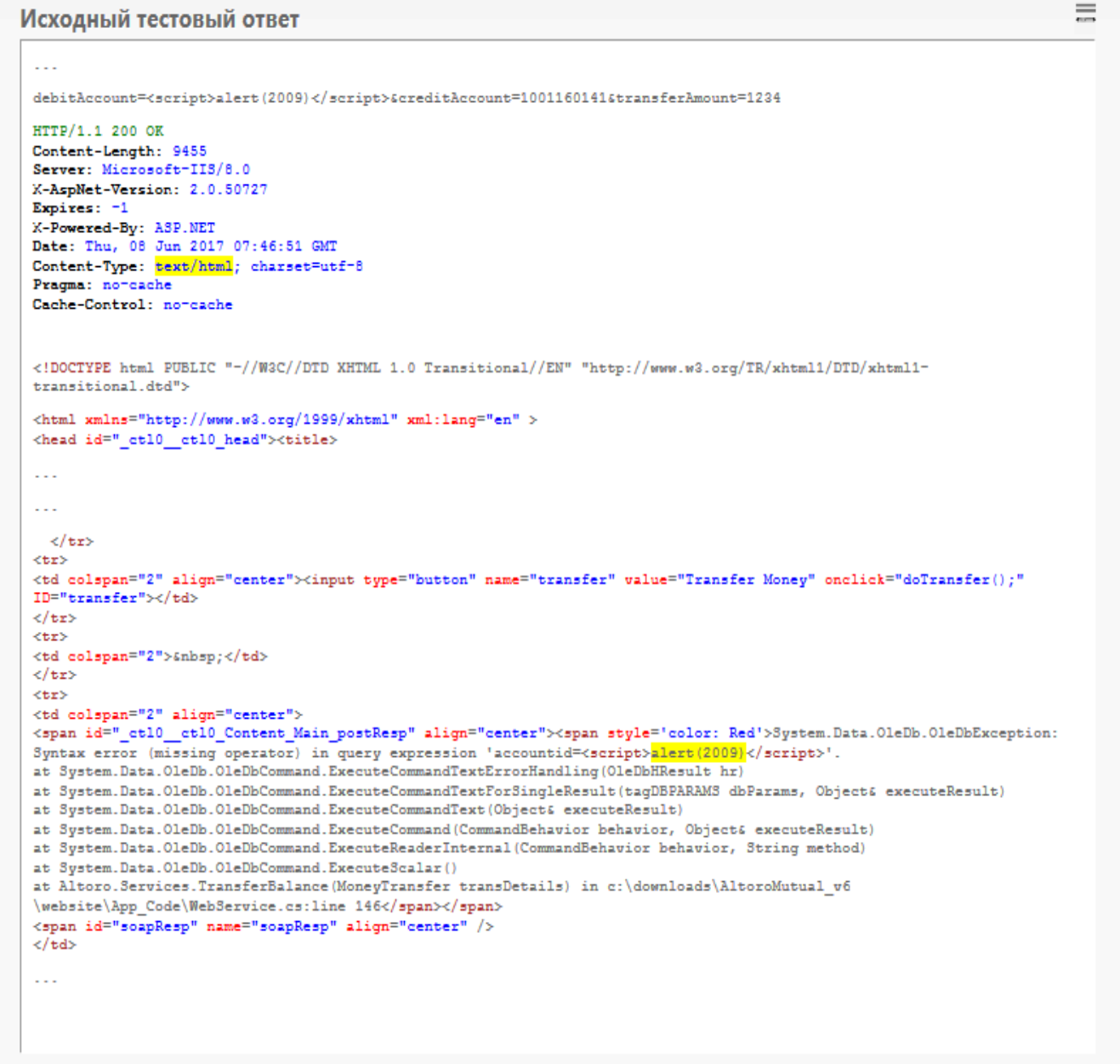

Рисунок 4.10 Тестова відповідь

Коментарі з можливими причинами виникнення дефекту, технічним описом, різновидами данної вразливості зображено на рисунку 4.11.

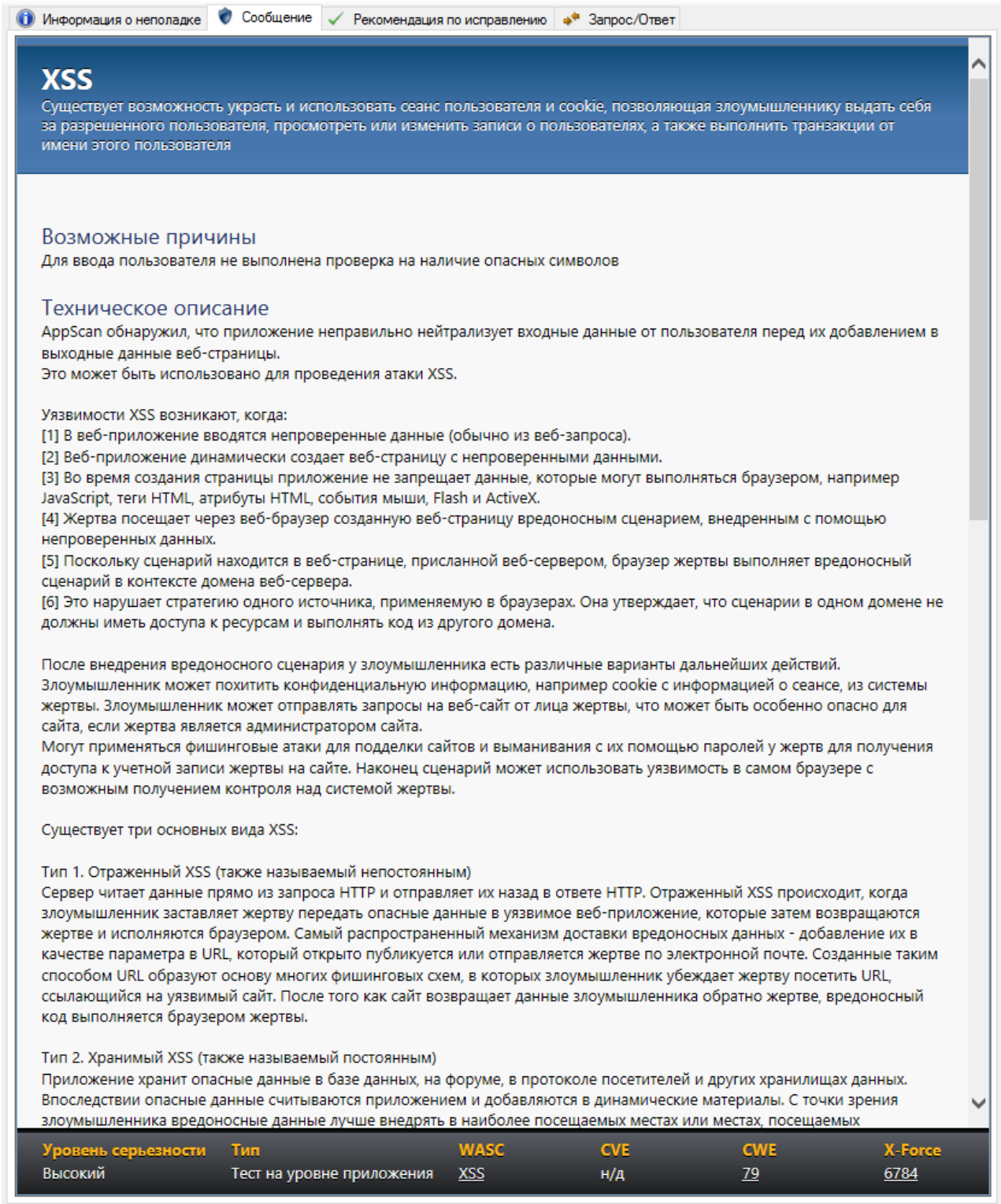

Рисунок 4.11 Можливі причини виникнення вразливості

Та поради щодо способів виправлення вихідного коду з метою усунення вразливості (рисунок 4.12):

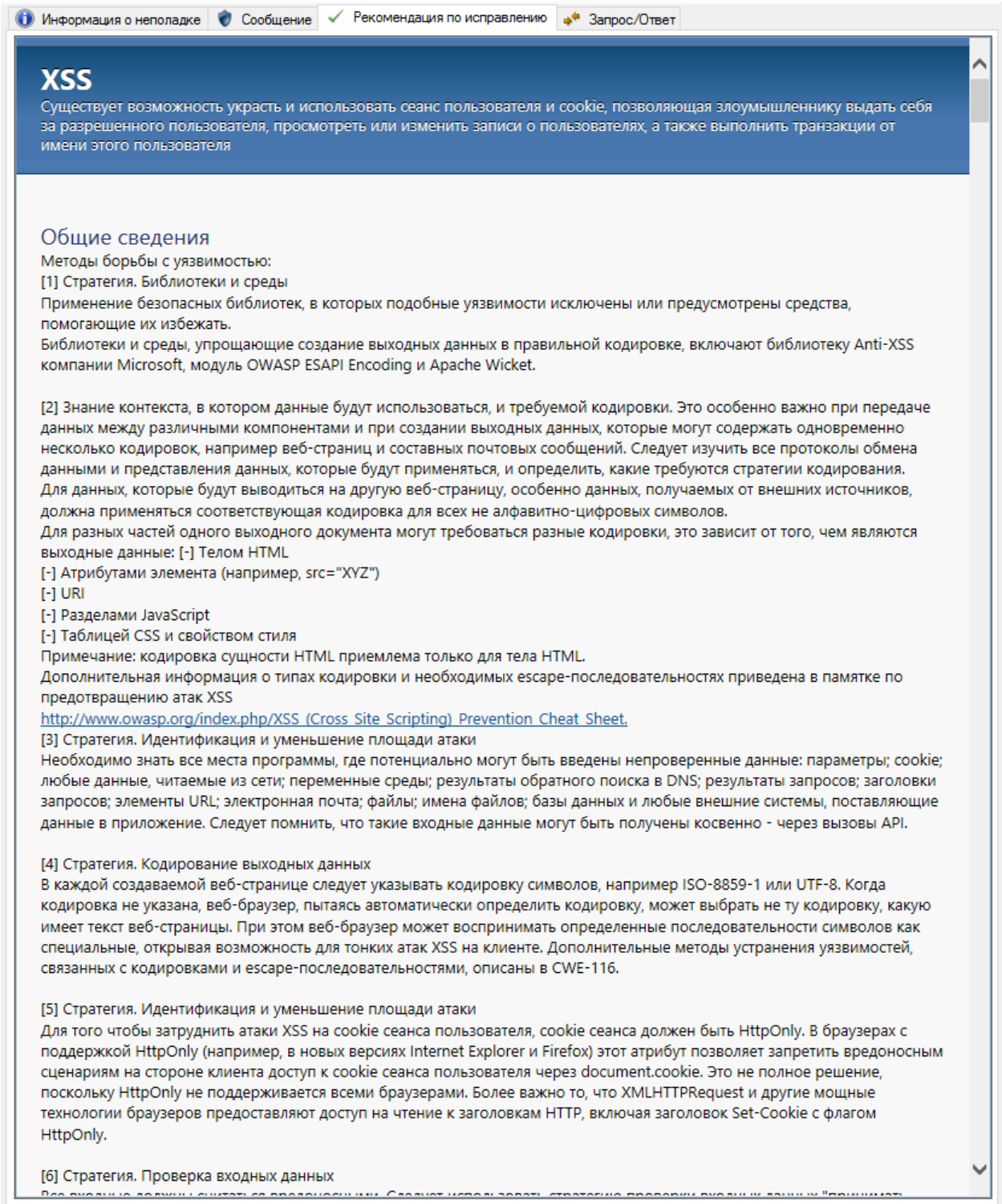

Рисунок 4.12 Загальні відомості про вразливість

Також в AppScan є можливість автоматичної генерації звіту по результатам тестування.

Для прикладу обрано сторінку з описом знайдених вразливрстей та їх кількості (рисунок 4.13):

Сводка

Типы неполадок 35

TOC

	Тип неполадки	Количество проблем
В	XSS	9
В	XSS на основе DOM	1
В	Внедрение SQL	13
В	Внедрение SQL вслепую	1
В	Обход идентификации путем внедрения SQL	2
В	Предсказуемые идентификационные данные для входа в систему	1
В	Ядовитый нулевой байт - получение файлов Windows	1
С	CSRF	9
С	В cookie зашифрованного сеанса (SSL) отсутствует атрибут Secure	5
С	Внедрение XPath	1
С	Внедрение ссылки (для CSRF)	5
С	Не обновляется идентификатор сеанса	1
С	Неадекватная блокировка учетной записи	1
С	Разбиение ответа HTTP	1
С	Фишинг с использованием фреймов	5
Н	В cookie сеанса отсутствует атрибут HttpOnly	1
Н	В поле пароля не отключен HTML-атрибут autocomplete	3
Н	Найден шаблон ошибки базы данных	20
Н	Найдена страница SSL с возможностью сохранения в кэше	14
Н	Незашифрованный параметр __VIEWSTATE	4
Н	Неподписанный параметр __VIEWSTATE	4
Н	Отсутствует заголовок "Content-Security-Policy"	5
Н	Отсутствует заголовок "X-Content-Type-Options"	5
Н	Отсутствует заголовок "X-XSS-Protection"	5
Н	Отсутствует заголовок HTTP Strict-Transport-Security	5
Н	Отсутствует защита от межфреймовых внедряемых сценариев	5
Н	Параметр Query в запросе SSL	8

Рисунок 4.13 Приклад сторінки звіту

Висновки

Даний розділ було присвячено виконанню тестування веб-додатка на наявність в ньому вразливостей в системі безпеки. Для цього було використано програмний пакет IBM Rational AppScan. В результаті було відшукано ряд серйозних вразливостей, які у випадку не виправлення, можуть стати місцем куди буде націлено атаку на додаток.

ЗАГАЛЬНІ ВИСНОВКИ

В ході роботи було обґрунтовано актуальність дослідження методів забезпечення якості програмних додатків, заснованих за використанні статистичних методів контролю

Проведено аналіз моделей життєвого циклу розробки програмних додатків. Дано визначення різних складових якості відповідно до стандарту ISO 9126:2001.

Досліджено вимоги до безпеки для різних компонент розроблюваного програмного додатку, такі як модуль аутентифікації, модулі шифрування та управління сеансами, а також до архітектури в цілому.

Проведено аналіз методів забезпечення якості на основі статичних планів контроля якості, які при їх використанні в умовах сучасного ринку IT дають змогу забезпечити високу якість і при цьому зменшити витрати на ресурси.

Використовуючи IBM Rational AppScan було закладено план контролю та отримано список вразливостей, що дозволило проаналізувати якість програмного додатку, який було досліджено. Використання таких комплексів дає можливість вивести якість додатків на доволі високий рівень. Це можливо, адже така система акумулює в собі набагато більше знань в конкретній області аніж невелика група людей.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO 8402-94 Управління якістю та забезпечення якості- Словник.
2. ISO 9000-2000 Системи менеджменту якістю. Основні положення та словник.
3. ISO 9126-1 Оцінка програмних продуктів. Характеристики якості та поради по їх застосуванню Основні положення та словник.
4. OWASP ASVS 3.0. Стандарт верифікації захищеності додатків.
5. Миттаг Х.-Й., Ринне Х. Статистические методы обеспечения качества. – М.: Машиностроение, 1995. –39-44 с.,76-83 с., 92 с.,143-170 с., 206- 215 с.,272-294 с.,307-327 с., 334-466 с..
6. СОУ 64.2 – 00017584 – 005:2009 “Телекомунікаційні мережі передавання даних загального користування. Система показників якості послуг передачі даних та доступу до інтернету. Загальні положення”
7. Яглом И.М. Математические структуры и математическое моделирование. М.: Сов.радио, 1980.
8. Яшин А.М. Разработка экспертных систем. Д.: ЛПИ, 1990.
9. Boehm B. Seven Basic Principles of Software Engineering. Infotech State of the Art Report on Software Engineering, 1977.
10. Лысаковский В.А., Журавлева Э.М., Амарян М.Р. Критерии оценки эффекта от автоматизации управления
11. Ефимов В.В. Статистические методы в управлении качеством Ульяновск: УлГТУ, 2003
12. Богатырев А.А., Филиппов Ю.Д. Стандартизация статистических методов управления качеством. М., Изд-во "Стандартов", 1989 г.
13. Бендерский А.М., Богатырев А.А., Баумгартен А.В. Стандартизация статистических методов управления качеством. М., Изд-во "Стандартов", 1983 г.
14. Богатырев А.А. /статья в сборнике/. Построение и функционирование систем управления качеством продукции, стр.140-154 "Организация

внедрения статистических методов контроля качества продукции на предприятиях". М., Изд-во "Стандартов", 1978 г.

15. Боровков А.А. Математическая статистика. Оценка параметров. Проверка гипотез. М., Изд-во Наука, 1984 г.
16. Хэнсен Б. Контроль качества. М., Прогресс, 1968 г.
17. Адлер Ю.П., Розовский Б.Л. Оперативное статистическое управление качеством. М., Знание, I984 г.
18. Хаммельблау Д. Анализ процессов статистическими методами. М., Мир, 1973 г.