Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління _____
(повна назва)

Кафедра _____Безпеки інформаційних технологій _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

__Методика оцінки сукупних ризиків безпеки при розробці веб-додатку_____
(тема)

Виконав:

студент __2_ курсу, групи __БІКСм-20-1_____

____Кочанов М.О._____

(прізвище, ініціали)

Спеціальність _____125 Кібербезпека _____

(код і повна назва спеціальності)

Освітня програма ___«Безпека інформаційних
_____і комунікаційних систем»____
(повна назва освітньої програми)

Керівник __доцент Федюшин О.І._____

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____Халімов Г.З.__
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Безпеки інформаційних технологій_____

Рівень вищої освіти_____ другий (магістерський)_____

Спеціальність__125 Кібербезпека _____
(код і повна назва)

Освітня програма_____«Безпека інформаційних і комунікаційних систем»___
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Кочанову Максиму Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ____ Методика оцінки сукупних ризиків безпеки при розробці веб-додатку_____
затверджена наказом по університету від «08» листопада 2021 р. № 1685Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10.12.2021 р.
3. Вихідні дані до роботи _ статистичні дані щодо кіберінцидентів пов'язаних з веб додатками_____
_____
_____
_____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі _____
Загрози безпеці веб додатків.
Методологія аналізу загроз безпеки.
Аналіз загроз безпеки.
Опис захисту від загроз за НД ТЗІ 2.5-010-03.
Розробка рекомендацій команді для забезпечення безпеки.
Реалізація тестування безпеки веб додатку.
Опис винятків з загальної методології.
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _презентаційний матеріал у вигляді слайдів _____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вибір здобувачем теми кваліфікаційної роботи	02.09.2021	Виконано
2	Затвердження плану і завдання кваліфікаційної роботи	09.11.2021	Виконано
3	Аналіз завдання, пошук та аналіз літературних джерел за темою роботи	10.11.2021-18.11.2021	Виконано
4	Виконання кваліфікаційної роботи	19.11.2021-30.11.2021	Виконано
5	Оформлення пояснювальної записки	01.12.2021-12.12.2021	Виконано
6	Здача на перевірку та підпис кваліфікаційної роботи керівнику	13.12.2021	Виконано
7	Проходження перевірки на плагіат та нормоконтроль кваліфікаційної роботи	15.12.2021	Виконано
8	Допуск завідувачем кафедри до захисту кваліфікаційної роботи	15.12.2021	Виконано
9	Захист кваліфікаційної роботи	17.12.2021	Виконано

Дата видачі завдання _____ _____20___ р.

Студент _____
(підпис)

Керівник роботи _____ _____ доцент Федюшин О.І._
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи містить : 90 с., 21 табл., 1 рис., 17 джерел.

ВЕБ ДОДАТОК, СПИСОК ЗАГРОЗ, РИЗИК, МЕТОДОЛОГІЯ АНАЛІЗУ, КОМЛПЕКС РЕКОМЕНДАЦІЙ, ТЕСТ ПЛАН, ТЕСТОВА СТРАТЕГІЯ

Об’єкт дослідження – забезпечення захисту веб додатків.

Предмет дослідження – загрози безпеки для веб додатків.

Мета роботи – сформувати ефективний комплекс рекомендацій для команди розробки веб додатку задля забезпечення захисту додатку від найбільш небезпечних загроз безпеці на основі методики оцінки сукупних ризиків безпеки.

Методи дослідження – аналіз статистичної інформації щодо загроз безпеці веб додаткам з урахуванням методики оцінки сукупного ризику та формування комплексу рекомендацій з розробки захищеного веб додатку.

Надані результати аналітичного та статистичного дослідження загроз веб додаткам. На основі отриманих результатів відібрані найбільш поширені загрози. Надано результати аналізу методики оцінки ступеня небезпеки вразливостей, яка заснована на методиці оцінки ризиків OWASP. Для кожної з розглянутих загроз надається загальна інформація про ймовірність її виникнення та можливі технічні наслідки, що отримана з використанням методики оцінки ризиків. Розглянута та проаналізована реалізація захисту веб додатків згідно з НД ТЗІ 2.5-010-03 . Надано пропозиції щодо протидії найбільш небезпечним загрозам веб додаткам для організації. Сформовано комплекс рекомендацій для розробників щодо забезпечення безпеки веб додатку . Описано винятки з загального випадку.

ABSTRACT

Explanatory note to the qualification work includes: 90 pages, 21 tables, 1 figure, 17 sources.

WEB APPLICATION, LIST OF THREATS, RISK, METHODOLOGY OF ANALYSIS, COMPLEX OF RECOMMENDATIONS, TEST PLAN, TEST STRATEGY

The object of research is protection of web applications.

The subject of the research is security threats for web applications.

The purpose of the work is to form an effective set of recommendations for the web application development team to ensure the protection of the application from the most dangerous security threats based on methodology of total risk assessment .

Research methods - analysis of statistical information on security threats to web applications taking into account the methodology of total risk assessment and formation of a set of recommendations for the development of a secure web application.

The results of analytical and statistical study of threats for web applications were presented. Based on the obtained results, the most common threats were selected. The results of the risk analysis for selected threats were presented. For each of the considered threats, general information on the probability of its occurrence and possible technical consequences was provided using the risk assessment methodology. The implementation of web application protection in accordance with ND TZI 2.5-010-03 was considered and analyzed. Suggestions for counteracting the most dangerous threats to web applications for the organization were provided. A set of recommendations for developers to ensure the security of the web application was presented. Exceptions to the general case were describe

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

URL - Uniform Resource Locator

OWASP - Open Web Application Security Project

SQL - Structured Query Language

NoSQL - Not Only SQL

LDAP - Lightweight Directory Access Protocol

API - Application Programming Interface

XML - Extensible Markup Language

URI - Uniform Resource Identifier

HTTP - HyperText Transfer Protocol

HTML - HyperText Markup Language

XSS - Cross-Site Scripting

XXE - Xml eXternal Entity

CMS - Content Management System

CVE - Common Vulnerabilities and Exposures

SFA - Single-factor Authentication

2FA - Two-Factor Authentication

MFA - Multi-Factor Authentication

TLS - Transport Layer Security

SSRF - Server-side request forgery

DTD - Document Type Definition

DAST - Dynamic Application Security Testing

SAST - Static Application Security Testing

SOAP - Simple Object Access Protocol

CSP - Content Security Policy

ПЗ - програмне забезпечення

ІТ - інформаційні технології

ВСТУП

Останнім часом широкого поширення набув список сервісів та ресурсів, які зародилися в Інтернеті. Інтернет перетворився з одноманітних статичних сторінок на потужний інструмент для взаємодії та спілкування з кінцевими користувачами. Веб-програми зараз набирають безпрецедентної популярності, тому що вони пропонують багато важливих переваг, яких не вистачає звичайним додаткам. Ці переваги включають:

- встановлення веб-додатків дешевше та набагато простіше;
- оновлювати веб-додатки вимагає менше зусиль;
- веб-додатки стали більш гнучкими та практичними для кінцевого користувача;
- веб-додатки спрощують організацію зберігання даних.

Завдяки цим перевагам багато компаній зараз фокусують свої зусилля саме на розробці веб додатків. Але ці компанії не беруть до уваги той факт, що підключення до Інтернету робить їх більш доступними не тільки для звичайних користувачів, але і для злочинців.

Кіберзлочинність наразі більш розвинута, ніж будь-коли – майже кожна компанія має свій власний веб-сайт, і онлайн-зловмисник може легко залишатися повністю анонімним. За статистикою на 2020 рік – 30% усіх кібератак було здійснено у веб-сегменті корпорацій, 50% на інфраструктуру, решта – на користувачів та мобільні додатки.

Задля більшої реалізації основних принципів конфіденційності, цілісності, аутентифікації, авторизації, доступності і незаперечності, тобто для створення цільної системи безпеки, організації часто витрачають великі гроші на тестування безпеки. Цей процес займає багато часу і може не виправдати інвестицій. Наприклад, для невеликих компаній, що працюють за запитами закордонних замовників , часто неможливо найняти професіоналів та провести

повне тестування продукту. В рамках бюджету компаніям краще визначити основні загрози для свого продукту та протидіяти їм.

З цієї причини в даній роботі було зібрано інформацію про найбільш поширені загрози для веб-додатків та розроблено методологію аналізу цих загроз для виявлення найважливіших з них. Також в роботі представлено комплекс рекомендацій щодо розробки захищеного веб додатку.

1 ЗАГРОЗИ БЕЗПЕЦІ ВЕБ ДОДАТКІВ

Веб-додаток - клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Логіка веб-додатки розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є кросплатформовими службами.

Так як існує велика кількість різноманітних загроз для веб додатків та методів для захисту від даних загроз, у даній роботі проводиться дослідження з метою виявлення найпоширеніших загроз, їх аналізу та розробці методів захисту саме від них, адже реалізувати повний комплекс захисту від усіх можливих атак на веб додаток неможливо через обмеження у фінансах та часі.

Отже, найбільш поширеними загрозами для безпеки веб додатків є загрози приведені в таблиці 1.1, що були відібрані на підставі досліджень OWASP (Open Web Application Security Project)[1] і аналізу статистики IT компаній.

Таблиця 1.1 – Загрози безпеці веб додатків

Загроза	Опис
Ін'єкція	Уразливості, пов'язані, наприклад, з ін'єкціями SQL, NoSQL, OS і LDAP, виникають, коли неперевірені дані відправляються інтерпретатору в складі команди або запиту. Шкідливі дані можуть змусити інтерпретатор виконати непередбачувані команди або звернутися до даних без проходження відповідної авторизації.

Продовження таблиці 1.1

Недоліки аутентифікації	Функції додатків, пов'язані з аутентифікацією і управлінням сесіями, часто некоректно реалізуються, дозволяючи зловмисникам скомпрометувати паролі, ключі або сесійні токени, а також експлуатувати інші помилки реалізації для тимчасового або постійного перехоплення облікових записів користувачів.
Розголошення конфіденційних даних	Багато веб-додатків та API мають поганий захист критичних фінансових, медичних або персональних даних. Зловмисники можуть викрасти або змінити ці дані, а потім здійснити шахрайські дії з кредитними картами або персональними даними. Конфіденційні дані вимагають додаткових заходів захисту, наприклад їх шифрування при зберіганні або передачі, а також спеціальних запобіжних заходів при роботі з браузером.
Зовнішні сутності XML (XXE)	Старі або погано налаштовані XML-процесори обробляють посилання на зовнішні сутності всередині документів. Ці сутності можуть бути використані для доступу до внутрішніх файлів через обробники URI файлів, загальні папки, сканування портів, віддалене виконання коду і відмову в обслуговуванні.
Недоліки контролю доступу	Дії, дозволені автентифікованим користувачам, часто некоректно контролюються. Зловмисники можуть скористатися цими недоліками і отримати несанкціонований доступ до облікових записів інших користувачів або конфіденційної інформації, а також змінити призначені для користувача дані або права доступу.

Продовження таблиці 1.1

Некоректна настройка параметрів безпеки	Некоректна настройка безпеки є поширеною помилкою. Це відбувається через використання стандартних параметрів безпеки, неповної або специфічною настройки, відкритого хмарного зберігання, некоректних HTTP-заголовків і докладних повідомлень про помилки, що містять критичні дані. Всі ОС, фреймворки, бібліотеки і додатки повинні бути не тільки налаштовані належним чином, але і своєчасно коректуватися і оновлюватися.
Міжсайтове виконання сценаріїв (XSS)	XSS має місце, коли додаток додає неперевірені дані на нову веб-сторінку без їх відповідної перевірки або перетворення, або коли оновлює відкриту сторінку через API браузера, використовуючи надані вами дані, що містять HTML або JavaScript код. За допомогою XSS зловмисники можуть виконувати сценарії в браузері жертви, що дозволяють їм перехоплювати сесії користувачів, підміняти сторінки сайту або перенаправляти користувачів на шкідливі сайти.
Небезпечна десеріалізація	Небезпечна десеріалізація часто призводить до віддаленого виконання коду. Помилки десеріалізації, що не призводять до віддаленого виконання коду, можуть бути використані для атак з повторним відтворенням, впровадженням і підвищенням привілеїв.

Продовження таблиці 1.1

Використання компонентів з відомими вразливостями	Компоненти, такі як бібліотеки, фреймворки і програмні модулі, запускаються з привілеями програми. Експлуатація уразливого компонента може привести до втрати даних або перехоплення контролю над сервером. Використання додатками і API компонентів з відомими уразливими порушити захист додатку і привести до серйозних наслідків.
Недоліки журналювання і моніторингу	Недоліки журналювання і моніторингу, а також відсутність або неефективне використання системи реагування на інциденти, дозволяє зловмисникам розвинути атаку, приховати свою присутність і проникнути в інші системи, а також змінити, перезавантажити або знищити дані. Проникнення в систему зазвичай виявляють тільки через 200 днів і, як правило, сторонні дослідники, а не в рамках внутрішніх перевірок або моніторингу.

2 МЕТОДОЛОГІЯ АНАЛІЗУ ЗАГРОЗ БЕЗПЕКИ

2.1 Поняття ризику

Як не існує ідентичних організацій, так і немає ідентичних зловмисників, цілей та наслідків атак. Якщо організація використовує систему управління контентом (CMS) для публікації новин, а інша медична організація використовує ту саму систему для зберігання медичних даних, загрози та ризики для цих організацій будуть дуже різними. Важливо визначати ризики для організації на основі загроз, які можуть настати у ході діяльності даної організації.

Ризик завжди передбачає імовірнісний характер результату, і даний термін часто сприймається як можливість отримання несприятливого результату (збитку), хоча його можна описати також як можливість отримання результату, відмінного від очікуваного. У цьому сенсі стає можливим говорити як про ризик втрат, так і про ризик надприбутку.

У фінансових колах ризик - це поняття, що стосується очікувань людей від подій. Тут це може вказувати на потенційно небажаний вплив на актив або його характеристики, який може бути результатом будь-якої минулої, теперішньої чи майбутньої події. У повсякденному використанні ризик часто використовується як синонім ймовірності втрати чи загрози.

У професійних оцінках ризику , ризик зазвичай поєднує у собі ймовірність події з впливом, який він міг би надати, і навіть з обставинами, які супроводжують наступ цієї події.

В IT індустрії розмірковуючи про поняття ризику, слід інтерпретувати його як імовірність зазнати фінансових чи репутаційних збитків при роботі з програмним забезпеченням.

2.2 Методи визначення ризику для організації (користувача)

Зловмисники можуть завдати шкоди бізнесу або організації, використовуючи веб додаток. Такі способи використання зловмисником веб програми є дуже серйозною загрозою для користувача цієї програми, що може призвести до негативних наслідків.

Уразливості безпеки веб додатків можна в деяких випадках легко виявити та використовувати, а в інших - важко виявити навіть сам факт уразливості системи безпеки. Можливі збитки від використання вразливостей безпеки у веб додатку також можуть змінюватись від незначних фінансових втрат до повного банкрутства компанії.

Щоб визначити ризики вразливості безпеки веб додатку певної організації, слід оцінити ймовірності, пов'язані з джерелами загроз, векторами атак і вразливістю безпеки, а потім об'єднати їх з оцінкою технічної та репутаційної шкоди даній організації [3]. На малюнку 2.1 показано метод визначення ризику.

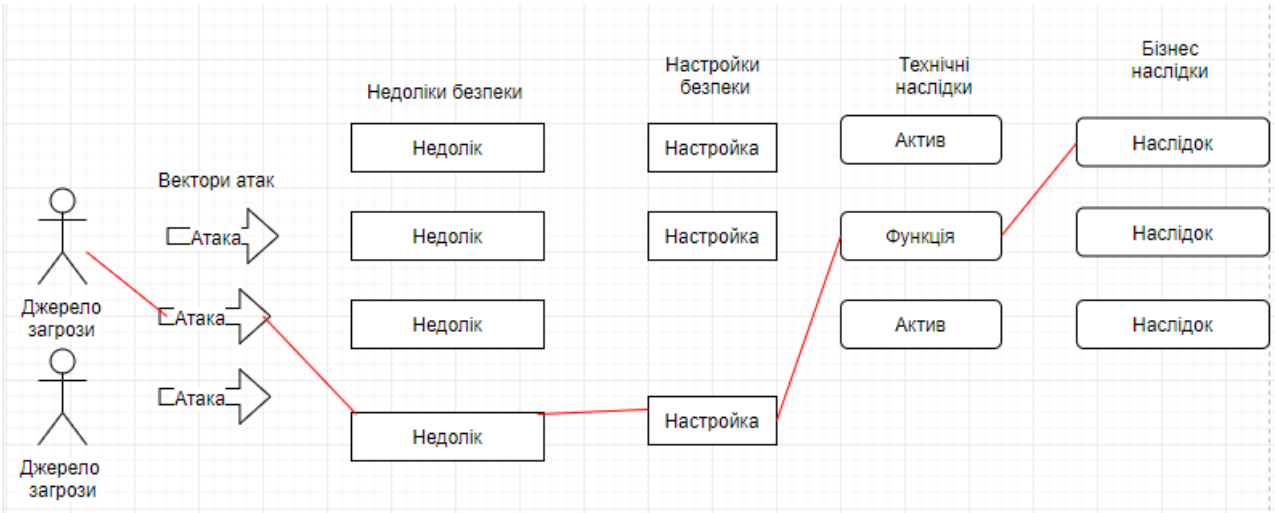

Рисунок 2.1 – Метод визначення ризику

2.3 Методика оцінки ризиків OWASP

Методика оцінки сукупного ризику, розроблена у даній роботі, ґрунтується на методиці оцінки ризику OWASP [4].

Методика оцінки ризиків OWASP для кожної категорії загроз оцінює недоліки, характерні для стандартного веб додатку, на основі їх факторів ймовірності.

Методику оцінки ризиків OWASP можна представити у вигляді алгоритму:

1. Для кожної загрози визначається ступінь її поширеності (за статистикою) та її фактор небезпеки (технічні наслідки).

2. На основі опису загроз у CVE (Common Vulnerabilities and Exposures) для кожної загрози визначається складність виявлення та експлуатації.

3. На підставі отриманих даних рівень критичності кожного фактора класифікується від 1 (низький) до 3 (високий).

4. Проаналізовані загрози формуються у список з числовим значенням фактору поширеності.

До переваг наведеної методики можна віднести той факт, що вона надає узагальнені дані про популярні загрози, але не інформацію про конкретні уразливості в реальних додатках та API. Отже, ніхто, крім власника або менеджера програми, не зможе точно оцінити ризики, що загрожують певній програмі. Тільки власник має найбільш повні знання для оцінки критичності ваших даних, наявності можливих загроз, а також принципів роботи та використання вашої системи.

Основним недоліком цієї методики слід вважати сильну орієнтацію на факторі поширеності загрози, оскільки існують досить поширені загрози, які мають слабкі технічні наслідки , тому повинні мати більш низький пріоритет при розробці захисту.

2.4 Методика оцінки сукупного ризику

Методика оцінки сукупного ризику - це вдосконалена методика оцінки ризиків OWASP, яка, на відміну від існуючої методики, дозволяє оцінювати загрози для веб додатків не тільки за фактором їх поширення , але й за їх сукупним ризиком для програми.

Методику оцінки сукупного ризику можна представити у вигляді алгоритму:

1) Для кожної загрози визначається ступінь її поширеності (за статистикою) та її фактор небезпеки (технічні наслідки).
2) На підставі опису загрози у CVE визначається складність виявлення та експлуатації для кожної загрози.
3) На підставі отриманих даних рівень критичності кожного фактора класифікується від 1 (низький) до 3 (високий).
4) Значення коефіцієнтів поширеності, складності виявлення та складності експлуатації складаються та множаться на значення технічних наслідків.
5) Отримані дані заносяться до нової категорії - сукупний ризик.
6) Проаналізовані загрози формуються в список за фактором сукупного ризику.

Запропонований підхід також не бере до уваги джерела загроз та технічні характеристики окремих додатків, але формує пріоритет для розробки захисту від загрози, що ґрунтується не на її поширеності, а на загальному ризику для додатка.

У таблиці 2.1 наведена зведена інформація щодо алгоритму методики оцінки сукупного ризику.

Таблиця 2.1 – Зведені інформація щодо методики оцінки сукупного ризику

Джерела загрози	Складність експлуатації (F)	Поширеність уразливості (A)	Складність виявлення (C)	Технічні наслідки (X)	Сукупний ризик
Залежить від додатку	Просто: 3	Дуже поширена :3	Просто: 3	Важкі: 3	$(F + A + C) \cdot X$
	Середньо: 2	Поширена: 2	Середньо: 2	Помірні: 2	
	Складно: 1	Рідкісна: 1	Складно: 1	Незначні: 1	

3 АНАЛІЗ ЗАГРОЗ БЕЗПЕКИ ДЛЯ ВЕБ ДОДАТКІВ

На основі виявленого переліку найпопулярніших загроз для веб додатків і застосування удосконаленої методології сукупного ризику був проведений детальний аналіз даних загроз, а також були виявлені фактори ризику, характерні для кожної із загроз. Ці фактори визначалися на основі доступної статистики.

3.1 Ін'єкція

Ін'єкція [5] - це вразливість веб -безпеки, яка дозволяє зловмиснику втручатися у запити, які обробляє додаток . Як правило, це дозволяє зловмиснику переглядати дані, які вони зазвичай не можуть отримати. Це можуть бути дані, що належать іншим користувачам, або будь -які інші дані, до яких сама програма має доступ. У багатьох випадках зловмисник може змінювати або видаляти ці дані, викликаючи постійні зміни у вмісті або поведінці програми.

У деяких ситуаціях зловмисник може виконати атаку ін'єкції, щоб поставити під загрозу базовий сервер або іншу внутрішню інфраструктуру, або виконати атаку відмови в обслуговуванні.

Успішна атака ін'єкції може призвести до несанкціонованого доступу до конфіденційних даних, таких як паролі, дані кредитної картки або особисті дані користувача. Багато гучних порушень даних за останні роки стали результатом атаки SQL-ін'єкцій, що призвело до репутаційної шкоди та штрафних санкцій. У деяких випадках зловмисник може отримати постійний бекдор у системах організації, що призведе до довгострокового компромісу, який може залишатися непоміченим протягом тривалого періоду.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки: майже будь-яке джерело даних може виявитися вектором для ін’єкції: змінні оточення, параметри, зовнішні і внутрішні веб-

служби, а також всі типи користувачів. Ін’єкції стають можливими, якщо зловмисник може відправляти інтерпретатору шкідливі дані;

2) недоліки безпеки : ін’єкції особливо поширені в старому коді. Уразливості часто зустрічаються в SQL- LDAP-, XPathілі NoSQL-запитах, системних командах, XML обробниках, SMTP-заголовках, мовами виразів і ORM запитах. Впровадження легко виявити при аналізі коду. Сканери та фазери можуть допомогти зловмисникам знайти подібні уразливості;

3) наслідки : ін’єкції можуть призвести до втрати даних, їх пошкодження або розголошенню третім особам, а також до відмови в обслуговуванні. В деяких випадках контроль над вузлом може бути повністю перехоплений. Наслідки для бізнесу залежать від критичності додатки і даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.1.

Таблиця 3.1 – Зведена інформація про загрози ін’єкції

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Ін’єкція	Залежить від додатку	Просто: 3	Поширена : 2	Просто: 3	Важкі: 3	24

Визначити, що наш додаток вразливий до даної погрози можливо якщо:

- введені користувачем дані не перевіряються, не фільтруються або не очищається;
- динамічні запити або не параметрізовані виклики без контекстного екранування безпосередньо використовуються в інтерпретатор;
- шкідливі дані використовуються в пошукових параметрах об'єктно-реляційного відображення для вилучення додаткової, критичної інформації;

– шкідливі дані використовуються або додаються таким чином, що SQL код або команди містять структурні і шкідливі дані в динамічних запитах, командах або збережених процедурах.

Найбільш поширеними є SQL-, NoSQL-, ORM-, LDAP-, EL- або OGNL-ін’єкції, а також впровадження команд ОС. Те ж саме стосується всіх інтерпретаторів. Аналіз вихідного коду є найкращим способом виявлення впроваджень, за яким слід повне автоматизоване тестування всіх впроваджуються параметрів, заголовків, URL, cookie, JSON-, SOAP- і XML-даних.

Організації також можуть включати в процес безперервної інтеграції та розгортання ПЗ (CI / CD) статичне (SAST) і динамічне (DAST) тестування коду і додатків для виявлення нових вразливостей перед впровадженням програм в виробництво.

Наведемо приклади сценаріїв даного типу атак.

1) Додаток використовує недовірених дані при створенні наступного уразливого SQL-виклику: String query = "SELECT * FROM accounts WHERE custID = ' "+ request.getParameter (" id ") +"' ";

2) Беззастережна довіра додатків до фреймворку може привести до появи вразливих запитів (наприклад, в мові запитів HQL):
Query HQLQuery = session.createQuery ("FROM accounts WHERE custID = ' "+ request.getParameter (" id ") +"' ");

В обох випадках зловмисник змінює в своєму браузері значення параметра "id" для відправки 'or' 1 '=' 1. наприклад:
http://example.com/app/accountView?id= 'or' 1 '=' 1

Зміна обох запитів дозволяє отримати всі записи з таблиці облікових даних. Серйозніші атаки дозволяють змінити або видалити дані, а також викликати збережені процедури.

3.2 Недоліки аутентифікації

Аутентифікація [6] - це процес визначення того, чи є користувач тим, за кого він себе видає. Технологія автентифікації забезпечує контроль доступу до систем, перевіряючи, чи відповідають облікові дані користувача обліковим даним у базі даних авторизованих користувачів або на сервері аутентифікації даних. При цьому аутентифікація забезпечує безпеку системи, захищеність процесів та захист корпоративної інформації.

Існує кілька типів аутентифікації. З метою ідентифікації користувача, як правило, використовується ідентифікатор користувача, і автентифікація відбувається, коли користувач надає облікові дані, такі як пароль, що відповідає їх ідентифікатору користувача. Практика вимагання ідентифікатора користувача та пароля відома як однофакторна аутентифікація (SFA). В останні роки компанії посилили аутентифікацію, попросивши ввести додаткові фактори аутентифікації, такі як унікальний код, який надається користувачеві через мобільний пристрій при спробі входу або біометричний підпис, наприклад сканування особи або відбиток пальця. Дана практика відома як двофакторна автентифікація (2FA).

Фактори автентифікації можуть навіть виходити за межі SFA, яка вимагає ідентифікатора користувача та пароля, або 2FA, яка вимагає ідентифікатора користувача, пароля та біометричного підпису. Коли для автентифікації використовуються три чи більше факторів перевірки ідентичності - наприклад, ідентифікатор користувача та пароль, біометричний підпис та, можливо, особисте запитання, на яке повинен відповісти користувач, - це називається багатофакторною автентифікацією (MFA).

Аутентифікація дозволяє організаціям захищати свої мережі, дозволяючи лише аутентифікованим користувачам або процесам отримати доступ до їх захищених ресурсів. Це можуть бути комп’ютерні системи, мережі, бази даних, веб-сайти та інші мережеві програми чи послуги.

Аналізуючи недоліки аутентифікації, можна представити інформацію про даний тип загроз за наступними категоріями:

1) вектори атаки : зловмисники мають доступ до сотень тисяч дійсних комбінацій імен і паролів для атак на облікові записи, списки стандартних облікових даних адміністраторів, інструменти для автоматизації атак методом підбору і атак за словниками. Атаки на сесії добре вивчені, особливо в частині діючих токенів сесій;

2) недоліки безпеки : недоліки аутентифікації дуже поширені через виконання і реалізацію більшості засобів ідентифікації та контролю доступу. управління сесіями. Засіб управління сесіями є основою аутентифікації і контролю доступу та присутній у всіх додатках з контролем стану. Зловмисники можуть виявити недоліки аутентифікації вручну і експлуатувати їх, використовуючи автоматизовані інструменти, списки паролів і атаки за словником;

3) наслідки : для компрометації системи зловмисникові достатньо отримати доступ до кількох звичайних або одного адміністраторського облікового запису. Залежно від області використання програми результатом може стати відмивання грошей, шахрайство в сфері соціального забезпечення або крадіжка персональних даних, а також розголошення захищеної законом, конфіденційної інформації.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.2.

Таблиця 3.2 – Зведена інформація про загрози недоліків аутентифікації

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Недоліки аутентифікації	Залежить від додатку	Просто: 3	Поширена : 2	Середньо: 2	Важкі: 3	21

Підтвердження особистості користувача, аутентифікація і управління сесіями грають важливу роль в захисті від атак, пов'язаних з аутентифікацією.

Визначити, що наш додаток вразливий до даної погрози можливо якщо:

- допускається проведення автоматизованих атак, наприклад, на облікові записи, коли у атакуючого є список діючих імен і паролів користувачів;
- допускається проведення атак методом підбору або інших автоматизованих атак;
- допускається використання стандартних, ненадійних або добре відомих паролів, наприклад, "Password1" або "admin / admin";
- використовуються ненадійні або неефективні методи відновлення облікових даних і паролів, наприклад, "відповіді на основі знань", які є небезпечними;
- використовуються незашифровані, зашифровані або ненадійно хешовані паролі ;
- відсутня або є неефективною багатофакторна аутентифікація;
- відображаються ідентифікатори сесії в URL (наприклад, перезапис URL);
- не змінюються ідентифікатори сесій після успішного входу в систему;
- некоректно анулюються ідентифікатори сесій. Призначені для користувача сесії або токени аутентифікації (зокрема, токени єдиного входу (SSO)) неправильно анулюються при виході з системи або бездіяльності.

Наведемо приклади сценаріїв даного типу атак.

1) Атака на облікові записи, з використанням списків відомих паролів, є дуже поширеною. Якщо в додатку немає захисту від автоматизованих атак або атак на облікові записи, то воно може бути використано для визначення діючих облікових даних.

2) Більшість атак на аутентифікацію пов'язано з використанням виключно паролів. Вимоги , що раніше вважалися хорошими, до зміни пароля і його складності сприяють використанню повторному користувачами ненадійних

паролів. Організаціям рекомендується відмовитися від подібної практики та впровадити многофакторну аутентифікацію.

3) Тайм-аути сесій налаштовані некоректно. Люди використовують загальнодоступні комп'ютери для доступу до додатку, а замість "виходу з програми" просто закривають вкладку і йдуть. Зловмисник може відкрити той же самий браузер через годину і скористатися все ще діючою аутентифікацією користувача.

3.3 Розголошення конфіденційних даних

Останнім часом питання конфіденційної інформації викликає підвищену увагу, так як будь-яка інформація має цінність для її власника і потребує захисту від нецільового використання та розголошення іншим особам.

Закон України «Про інформацію» [7], зокрема р.2 ст. 21 під конфіденційною інформацією, має на увазі інформацію про суб'єкта, доступ до якої обмежено фізичною або юридичною особою, крім суб'єктів владних повноважень. Однак, ця інформація може поширюватися за згодою особи тільки відповідно до визначених нею умов, або в інших випадках, передбачених законом.

У свою чергу, ст. 505 Цивільного Кодексу України [8] закріплює визначення комерційної таємниці, тобто тієї інформації, яка вважається в цілому або в певній формі секретної, в зв'язку з чим має комерційну цінність і є предметом адекватним обставинам заходів щодо збереження її секретності, вжитих особою, яка її контролює.

Аналізуючи погрозу розголошення конфіденційних даних, можна представити інформацію щодо даної загрози за наступними категоріями:

1) вектори атаки : замість злому механізмів шифрування зловмисники крадуть ключі, проводять атаки за принципом "людина посередині" або отримують дані в незашифрованому вигляді з сервера в процесі їх передачі, або з клієнта користувача, наприклад, браузера. Подібні атаки зазвичай проводяться

вручну. Раніше отримані бази даних паролів можуть бути зламані методом підбору з використанням графічних процесорів ;

2) недоліки безпеки : протягом останніх років дана атака є найпоширенішою і небезпечною. Найчастіше зустрічається відсутність шифрування конфіденційних даних, а при наявності часто використовуються ненадійні алгоритми, протоколи, шифри, методи зберігання хешованих паролів або методи створення та управління ключами. Також легко виявити вразливості на стороні сервера для переданих даних, але не збережених ;

3) наслідки : через уразливості часто страждають всі персональні дані (медичні записи, облікові дані, дані кредитних карт), які повинні бути захищені згідно із законом, наприклад, в відповідно до Загального регламенту ЄС щодо захисту даних (GDPR) або локальними законами про недоторканність даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.3.

Таблиця 3.3 – Зведена інформація про загрози розголошення конфіденційних даних

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Розголошення конфіденційних даних	Залежить від додатку.	Середньо: 2	Дуже поширена : 3	Середньо: 2	Важкі: 3	21

Для перевірки чи є веб додаток уразливим до даного типу атаки треба дати відповідь на наступні питання:

- перш за все, необхідно визначити потрібний рівень захисту даних при їх передачі та зберіганні. Наприклад, паролі, номери кредитних карт, медичні записи, персональні дані і комерційні таємниці вимагають додаткового захисту, особливо якщо вони підпадають під дію закону про недоторканність даних або закону про захист фінансових даних;

- виконується чи ні шифрування переданих даних ? Це стосується протоколів передачі даних, таких як HTTP, SMTP і FTP. Особливо небезпечний зовнішній інтернет трафік;
- шифруються чи ні сховища критичних даних, а також резервні копії ?
- чи використовуються за замовчуванням або в більш ранніх версіях застарілі або ненадійні алгоритми шифрування?
- чи використовуються створені за замовчуванням, ненадійні або однакові шифроключі, а також чи застосовуються відповідні механізми контролю та зміни ключів?
- чи використовується шифрування, наприклад, присутні чи ні відповідні директиви безпеки користувальницьких агентів (браузерів) і заголовки?
- чи перевіряє призначений для користувача агент (напр. додаток або поштовий клієнт) дійсність отриманих сертифікатів?

Наведемо приклади сценаріїв даного типу атак.

1) Додаток шифрує номери кредитних карт в базі даних, використовуючи автоматичне шифрування БД. Однак ці дані автоматично розшифровуються при вилученні, дозволяючи за допомогою впровадження SQL-коду отримати дані кредитних карт в незашифрованому вигляді.

2) Сайт не використовує TLS для всіх сторінок або підтримує ненадійне шифрування. Зловмисник може переглянути мережевий трафік (наприклад, в небезпечної бездротової мережі), переключити з'єднання з HTTPS на HTTP, перехопити запити і викрасти сесійні cookie. Після цього він може використовувати отримані cookie для перехоплення сесії користувача (минулу аутентифікацію), змінивши особисті дані користувача. Також зловмисник може змінити всі передані дані, наприклад, одержувача грошового переказу

3.4 Зовнішні сутності XML (XXE)

Ін'єкція зовнішньої сутності XML (також відома як XXE) [9] - це вразливість веб-безпеки, яка дозволяє зловмиснику перешкоджати обробці XML-даних додатком. Це часто дозволяє зловмиснику переглядати файли у файловій системі сервера додатків та взаємодіяти з будь-якими внутрішніми або зовнішніми системами, до яких сама програма має доступ.

У деяких ситуаціях зловмисник може виконати атаку XXE, щоб скомпрометувати базовий сервер або іншу інфраструктуру внутрішнього середовища, використовуючи вразливість XXE для виконання атак із підробкою запитів на стороні сервера (SSRF).

Деякі програми використовують формат XML для передачі даних між браузером і сервером. Додатки, які роблять це, практично завжди використовують стандартну бібліотеку або API платформи для обробки даних XML на сервері. Уразливості XXE виникають через те, що специфікація XML містить різні потенційно небезпечні функції, а стандартні парсери підтримують ці функції, навіть якщо вони зазвичай не використовуються програмою.

Зовнішні сутності XML - це тип користувацької сутності XML, визначені значення якої завантажуються поза межами DTD, у якому вони оголошені. Зовнішні сутності особливо цікаві з точки зору безпеки, оскільки дозволяють визначати сутність на основі вмісту шляху до файлу або URL -адреси.

Існують різні типи атак XXE:

- використання XXE для отримання файлів, де визначається зовнішня сутність, що містить вміст файлу, і повертається у відповіді програми;
- використання XXE для виконання атак SSRF, де зовнішня сутність визначається на основі URL-адреси для внутрішньої системи;
- експлуатація сліпого XXE виводить дані поза діапазону, де чутливі дані передаються від сервера додатків до системи, якою керує зловмисник;

– використання сліпого XXE для отримання даних за допомогою повідомлень про помилки, коли зловмисник може викликати повідомлення про помилку аналізу, що містить конфіденційні дані.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : зловмисники можуть експлуатувати вразливі обробники XML через завантаження XML або впровадження шкідливого контенту в XML-документи, використовуючи уразливий код, залежності або компоненти;

2) недоліки безпеки : за замовчуванням, більшість старих обробників XML дозволяють задавати зовнішні сутності, URI, які розіменовуються і обчислюються при обробці XML. Інструменти SAST дозволяють виявити вразливість шляхом перевірки залежностей і конфігурації. Інструменти DAST вимагають додаткових операцій, що виконуються вручну, для виявлення та експлуатації уразливості;

3) наслідки : подібні уразливості можуть використовуватися для отримання даних, виконання віддалених запитів з сервера, сканування внутрішньої системи, провокування відмови в обслуговуванні, а також здійснення інших атак. Наслідки для бізнесу залежать від критичності захисту всіх вразливих додатків і даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.4.

Таблиця 3.4 – Зведена інформація про загрози зовнішніх сутностей XML (XXE)

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Зовнішні сутності XML (XXE)	Залежить від додатку	Середньо: 2	Поширена : 2	Просто: 3	Важкі: 3	21

Додатки, особливо веб-служби або компоненти на основі XML, є уразливими в наступних випадках:

- додаток приймає XML безпосередньо або через вивантаження, особливо від недовірених джерел, або включає неперевірені дані в XML-документи, які потім обробляються XML-обробником;
- хоча б один з XML-обробників додатку або веб-служби на основі SOAP використовує визначення типу документів (DTD). Оскільки механізм відключення DTD залежить від обробника, рекомендується скористатися довідковою інформацією;
- додаток використовує SAML для ідентифікації в рамках федеративної безпеки або технології єдиного входу (SSO). SAML використовує XML для підтвердження ідентифікаторів, тому може бути уразливий;
- додаток використовує SOAP версії нижче 1.2. Воно може бути вразливе для XXE-атак, якщо XML-суті передаються фреймворку SOAP;
- якщо додаток вразливий для XXE-атак, то зловмисник може також викликати відмову в обслуговуванні або здійснити атаку з використанням мільйона XML-сутностей (Billion Laughs).

Наведемо приклади сценаріїв даного типу атак.

Було зафіксовано велику кількість XXE-атак, включаючи атаки на вбудовані пристрої. XXE виявляються в найбільш несподіваних місцях, включаючи глибоко вкладені залежності. Найпростішим способом реалізації атаки є завантаження (якщо підтримується) шкідливого XML-файла.

1) Зловмисник намагається отримати дані з сервера:

<? Xml version = "1.0" encoding = "ISO-8859-1"?>

<! DOCTYPE foo [

<! ELEMENT foo ANY>

<! ENTITY xxe SYSTEM "file: /// etc / passwd">]>

<Foo> & xxe; </ foo>

2) Зловмисник досліджує внутрішню мережу сервера, замінюючи вищевказану рядок ENTITY на:

<! ENTITY xxe SYSTEM "https://192.168.1.1/private">]>

3) Зловмисник намагається викликати відмову в обслуговуванні, використовуючи потенційно нескінченний файл:

<! ENTITY xxe SYSTEM "file: /// dev / random">]>

3.5 Недоліки контролю доступу

Контроль доступу [10] - це застосування обмежень щодо того, хто може виконувати спроби дій або отримувати доступ до ресурсів, які вони запросили. У контексті веб -додатків контроль доступу залежить від автентифікації та управління сеансами:

- аутентифікація ідентифікує користувача та підтверджує, що він той, ким він себе називає;
- керування сеансами визначає, які наступні запити HTTP здійснює той самий користувач;
- контроль доступу визначає, чи дозволено користувачеві виконувати дію, яку він намагається виконати.

Зламані засоби контролю доступу - це поширена і часто критична вразливість безпеки. Проектування та управління засобами контролю доступу є складною та динамічною проблемою, яка застосовує ділові, організаційні та юридичні обмеження до технічної реалізації.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : експлуатація контролю доступу є основним навиком зловмисників. Інструменти SAST і DAST можуть виявити відсутність контролю доступу, але не можуть перевірити працездатність при його наявності. Наявність контролю доступу можна виявити вручну, а його відсутність можна виявити автоматично в деяких фреймворками;

2) недоліки безпеки : уразливості, пов'язані з контролем доступу, досить

поширені через відсутність автоматичного виявлення і ефективного функціонального тестування розробниками. Контроль доступу зазвичай не перевіряється автоматичними статичними або динамічними тестами. Тестування вручну - найкращий спосіб виявлення відсутності або неефективності контролю доступу, включаючи методи HTTP (GET, PUT), контролери, прямі посилання на об'єкти та інше;

3) наслідки : виконання зловмисником дій з правами користувача або адміністратора; використання користувачем привілейованих функцій; створення, перегляд, оновлення або видалення будь-яких записів. Наслідки для бізнесу залежать від критичності захисту програми і даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.5.

Таблиця 3.5 – Зведена інформація про загрози недоліків контролю доступу

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Недоліки контролю доступу	Залежить від додатку	Середньо: 2	Поширена : 2	Середньо: 2	Важкі: 3	18

З точки зору користувача, елементи управління доступом можна розділити на наступні категорії:

- контроль вертикального доступу;
- контроль горизонтального доступу;
- контекстозалежні засоби контролю доступу.

Контроль вертикального доступу - це механізми, що обмежують доступ до чутливих функцій, недоступних для інших типів користувачів.

За допомогою вертикального контролю доступу різні типи користувачів мають доступ до різних функцій програми. Наприклад, адміністратор може

змінити або видалити обліковий запис будь -якого користувача, тоді як звичайний користувач не має доступу до цих дій. Контроль вертикального доступу може бути більш детальною реалізацією моделей безпеки, призначених для забезпечення політики бізнесу, такої як розподіл обов'язків та привілеїв.

Горизонтальні засоби контролю доступу - це механізми, які обмежують доступ до ресурсів лише тим користувачам, яким дозволено отримати доступ до цих ресурсів.

За допомогою горизонтального контролю доступу різні користувачі мають доступ до підмножини ресурсів одного типу. Наприклад, банківська програма дозволить користувачеві переглядати транзакції та здійснювати платежі зі своїх власних рахунків, але не з рахунків будь -якого іншого користувача.

Контекстозалежні елементи керування доступом обмежують доступ до функціональних можливостей та ресурсів залежно від стану програми або взаємодії користувача з нею.

Контекстозалежні елементи керування доступом запобігають виконанню користувачем дій у неправильному порядку. Наприклад, веб-сайт роздрібної торгівлі може перешкодити користувачам змінювати вміст кошика для покупок після здійснення платежу.

Контроль доступу передбачає наявність політики, яка визначає права користувачів. Обхід обмежень доступу зазвичай призводить до несанкціонованого розголошення, зміни або знищення даних, а також виконання непередбачених повноваженнями бізнес-функцій. Найбільш поширені уразливості контролю доступу включають:

- обхід обмежень доступу шляхом зміни URL, внутрішнього стану додатку або HTML-сторінки, а також за допомогою спеціально розроблених API;
- можливість зміни первинного ключа для доступу до записів інших користувачів, включаючи перегляд або редагування чужих облікових записів;

- підвищення привілеїв. Виконання операцій з правами користувача, який не входячи в систему, або з правами адміністратора, увійшовши в систему з правами користувача;
- маніпуляції з метаданими, наприклад, повторне відтворення або підміна токенів контролю доступу JWT або cookie-файлів, а також зміна прихованих полів для підвищення привілеїв чи некоректне анулювання JWT;
- несанкціонований доступ до API через некоректну настройки міждоменного використання ресурсів (CORS);
- доступ нерозпізнаних користувачів до сторінок, які вимагають аутентифікації, або доступ непривілейованих користувачів до привілейованим сторінок. Доступ до API з відсутнім контролем привілеїв для POST-, PUT- і DELETE-методів / запитів.

Загалом можна уникнути вразливостей контролю доступу, застосовуючи глибокий захист та реалізуючи наступні принципи:

- ніколи не покладайтесь лише на затуманення для контролю доступу;
- якщо ресурс не призначений для загальнодоступного доступу, забороніть доступ за замовчуванням;
- використовуйте єдиний загальнодоступний механізм для забезпечення контролю доступу;
- на рівні коду встановіть обов’язковість для розробників оголошувати доступ, дозволений для кожного ресурсу, і відмовляти у доступі за замовчуванням;
- ретельно перевіряйте засоби контролю доступу, щоб переконатися, що вони працюють як задумано.

Наведемо приклади сценаріїв даного типу атак.

1) Додаток використовує неперевірені дані в SQL-виклику, який звертається до інформації про обліковий запис:

pstmt.setString (1, request.getParameter ("acct"));

ResultSet results = pstmt.executeQuery ();

Зловмисник змінює в браузері параметр 'acct' для відправки бажаного номера облікового запису. Без належної перевірки атакуючий може отримати доступ до облікового запису будь-якого користувача.

http://example.com/app/accountInfo?acct=notmyacct

2) Зловмисник задає в браузері цільової URL. Для доступу до сторінки адміністрування потрібні права адміністратора.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Уразливість існує, якщо користувач без аутентифікації може отримати доступ до цих сторінок або якщо користувач без прав адміністратора може отримати доступ до сторінки адміністрування.

3.6 Некоректна настройка параметрів безпеки

Некоректна настройка параметрів безпеки [11] - це елементи безпеки, які неправильно налаштовані або залишаються незахищеними, піддаючи ризику ваші системи та дані. В принципі, будь -які погано задокументовані зміни конфігурації, налаштування за замовчуванням або технічні проблеми з будь - яким компонентом ваших кінцевих точок можуть призвести до неправильної конфігурації.

Помилки у конфігурації можуть статися з безлічі причин. Сучасна мережева інфраструктура надзвичайно складна і характеризується постійними змінами; організації можуть легко пропустити важливі параметри безпеки, включаючи нове мережеве обладнання, яке може зберігати стандартні конфігурації. Навіть якщо було створено безпечні конфігурації для кінцевих точок, слід часто перевіряти конфігурацію та засоби безпеки, щоб визначити неминучі відхилення конфігурації. Змінюються системи, в мережу вводиться нове обладнання, застосовуються виправлення - все це сприяє неправильній конфігурації.

Крім того, розробники можуть написати гнучкі правила брандмауера та створити мережеві ресурси для зручності під час створення програмного забезпечення та залишити їх без змін. Іноді адміністратори дозволяють зміни конфігурації для тестування або усунення несправностей і забувають повернути все до вихідного стану. Крім того, нерідкі випадки, коли працівники тимчасово вимикають свій антивірус, коли він замінює певні дії, наприклад, запустивши інсталятори, а потім забувають увімкнути його пізніше.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : зловмисники часто намагаються експлуатувати невиправлені уразливості, налаштовані за замовчуванням облікові записи, не використані сторінки, незахищені файли і каталоги для діставання несанкціонованого доступу або інформації про систему;

2) недоліки безпеки : налаштування безпеки може бути виконано некоректно на будь-якому рівні додатку, включаючи мережеві служби, платформи, веб-служби, сервер, базу даних, фреймворки, код, а також встановлені віртуальні машини, контейнери або сховища. Для пошуку вразливих налаштувань, налаштованих за умовчанням облікових записів, невикористовуваних служб, застарілих параметрів і подібного можна використовувати автоматизовані сканери;

3) наслідки : подібні уразливості дозволяють зловмисникам отримати несанкціонований доступ до системних даних або функцій, а також можуть привести до повної компрометації системи. Наслідки для бізнесу залежать від критичності захисту програми і даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.6.

Таблиця 3.6 – Зведена інформація про загрози некоректної настройки параметрів безпеки

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Некоректна настройка параметрів безпеки	Залежить від додатку	Просто: 3	Дуже поширена : 3	Просто: 3	Помірні: 2	18

Визначити, що ваш додаток вразливий до даної погрози можливо якщо:

- будь-який з компонентів програми недостатньо захищений або дозволи хмарних сервісів некоректно налаштовані;
- включені або присутні зайві функції (наприклад, невикористовувані порти, служби, сторінки, облікові записи або привілеї);
- облікові записи і паролі, створювані за замовчуванням, використовуються без змін;
- обробка помилок дозволяє здійснити трасування стека або отримати занадто докладні повідомлення про помилки;
- відключені або некоректно налаштовані останні оновлення безпеки;
- не вибрані безпечні значення параметрів захисту серверів додатків, фреймворків (наприклад, Struts, Spring, ASP.NET), бібліотек і іншого;
- сервер не використовує безпечні заголовки або директиви, а також якщо вони некоректно налаштовані;
- ПО застаріло або має уразливості;

Без організованої і регулярно виконуваної перевірки безпеки додатків системи більш схильні до ризику.

Наведемо приклади сценаріїв даного типу атак.

1) Сервер додатків поставляється зі зразками додатків, які залишаються на робочому сервері. Ці додатки містять відомі уразливості, що дозволяють зловмисникам скомпрометувати сервер. Якщо один з цих додатків є консоллю

адміністратора, а стандартні облікові записи не змінювалися, то атакуючий може увійти в додаток і перехопити контроль над ним, використовуючи стандартний пароль.

2) На сервері не натиснута кнопка вимкнення виведення списку файлів в каталогах, що дозволяє зловмисникові знайти і вивантажити скомпільовані Java класи, після декомпіляції і зворотного аналізу яких можна переглянути вихідний код. В результаті атакуючий може виявити уразливості і отримати доступ до додатку.

3) Сервер додатків налаштований на відправку докладних повідомлень про помилки, включаючи дані про трасування стека. Це може привести до розголошення важливої інформації, наприклад, про версії компонента, що містить відомі уразливості.

4) Постачальник хмарних послуг використовує стандартні дозволи загального доступу через інтернет для інших користувачів хмари. Це дозволяє отримати доступ до конфіденційної інформації, доступною в хмарному сховищі.

3.7 Міжсайтове виконання сценаріїв (XSS)

Міжсайтові сценарії (також відомі як XSS) [12] - це вразливість веб-безпеки, яка дозволяє зловмиснику поставити під загрозу взаємодію користувачів із вразливою програмою. Це дозволяє зловмиснику обійти політику погодження, яка призначена для відокремлення різних веб-сайтів один від одного. Вразливості міжсайтових сценаріїв зазвичай дозволяють зловмиснику маскуватися під користувача-жертву, виконувати будь-які дії, які користувач може виконати, і отримувати доступ до будь-яких даних користувача. Якщо користувач -жертва має привілейований доступ до програми, зловмисник може отримати повний контроль над усіма функціональними можливостями програми та даними.

Міжсайтові сценарії працюють, маніпулюючи вразливим веб-сайтом, щоб він повертав користувачам шкідливий JavaScript. Коли шкідливий код

виконується у веб-переглядачі жертви, зловмисник може повністю скомпрометувати свою взаємодію з додатком.

Існує три основних типи атак XSS. Це:

- відображений XSS, де шкідливий сценарій надходить із поточного запиту HTTP;
- збережений XSS, де шкідливий скрипт надходить із бази даних веб -сайту;
- XSS на базі DOM, де вразливість існує в коді на стороні клієнта, а не в коді на стороні сервера.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : автоматизовані інструменти можуть виявляти і експлуатувати всі три види міжсайтового виконання сценаріїв, більш того, фреймворки для їх експлуатації можна знайти у відкритому доступі ;

2) недоліки безпеки : міжсайтове виконання сценаріїв (XSS) є дуже поширеною уразливістю і виявляється в двох третинах усіх додатків. Автоматизовані інструменти можуть виявляти XSS автоматично, особливо в випадку опрацьованих технологій, таких як PHP, J2EE / JSP і ASP.NET;

3) наслідки : міжсайтове виконання сценаріїв буде мати наслідки середнього ступеня тяжкості в разі відбитого XSS або XSS на основі об'єктної моделі документа і серйозні наслідки в разі міжсайтового виконання збережених сценаріїв з віддаленим виконанням коду в браузері користувача, наприклад, крадіжка облікових даних, перехоплення сесій або установка шкідливого ПЗ.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.7.

Таблиця 3.7 – Зведена інформація про загрози міжсайтового виконання сценаріїв (XSS)

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Міжсайтове виконання сценаріїв (XSS)	Залежить від додатку	Просто: 3	Дуже поширена : 3	Просто: 3	Помірні: 2	18

Розглянемо детальніше типи XSS, що зазвичай експлуатуються в браузерах:

1) Відбите міжсайтове виконання сценаріїв (Reflected XSS): додаток або API включає неперевірені і не оброблені дані до складу HTML. Успішна атака може привести до виконання довільного HTML і JavaScript коду в браузері жертви. Зазвичай зловмисникові необхідно переконати користувача перейти по посиланню, що веде на шкідливу сторінку, наприклад, використовуючи атаку типу "водопій" або рекламу.

2) Міжсайтове виконання збережених сценаріїв (Stored XSS):додаток або API зберігає необроблені вхідні дані, з якими потім взаємодіють користувачі та адміністратори. Міжсайтове виконання збережених сценаріїв зазвичай вважається дуже небезпечною вразливістю.

3) Міжсайтове виконання сценаріїв на основі об'єктної моделі документа (DOM XSS): JavaScript фреймворки, односторінкові додатки і API динамічно додають шкідливі дані на сторінки, схильні до XSS на основі DOM. В ідеалі, програма не має відправляти шкідливі дані небезпечним JavaScript API.

Зазвичай XSS використовується для перехоплення сесій, крадіжки облікових записів, обходу MFA, заміни або підміни DOM-вузлів (напр. троянські панелі входу в систему), а також атак на браузери, наприклад, для завантаження шкідливого ПО, реєстрації натискань і інших атак на стороні клієнта.

Переважна більшість уразливостей XSS можна швидко та надійно знайти за допомогою веб-сканера вразливості Burp Suite.

Тестування вручну для відображеного та збереженого XSS зазвичай передбачає подання якогось простого унікального вводу (наприклад, короткого буквено-цифрового рядка) у кожну точку входу програми, визначення кожного розташування, де поданий вхід повертається у відповідях HTTP, та тестування кожного розташування окремо для визначення чи можна використовувати належним чином створений вхід для виконання довільного JavaScript. Таким чином, можливо визначити контекст, у якому відбувається XSS, і вибрати відповідне корисне навантаження для його використання.

Тестування вручну для XSS на основі DOM, що випливає з параметрів URL-адреси, передбачає подібний процес: розміщення простого унікального введення в параметрі, використання інструментів розробника веб-переглядача для пошуку в DOM цього введення та перевірка кожного розташування, щоб визначити, чи можна його використовувати. Однак інші типи DOM XSS важче виявити. Щоб знайти вразливості на основі DOM у вхідних даних, що не базуються на URL-адресах (наприклад, document.cookie), або в поглинаннях, що не базуються на HTML (наприклад, setTimeout), не існує заміни перегляду коду JavaScript, що може зайняти дуже багато часу. Веб-сканер вразливості Burp Suite поєднує статичний та динамічний аналіз JavaScript для надійної автоматизації виявлення вразливостей на основі DOM.

Одним з методів виявлення уразливостей XSS є політика безпеки вмісту.

Політика безпеки вмісту (CSP) - це механізм браузера, метою якого є пом'якшення впливу міжсайтових сценаріїв та деяких інших уразливостей. Якщо програма, яка використовує CSP, містить поведінку, подібну до XSS, тоді CSP може перешкоджати або запобігати використанню вразливості. Часто CSP можна обійти, щоб забезпечити використання основної вразливості.

Запобігання міжсайтовим сценаріям є тривіальним завдання у деяких випадках, але може значно поскладнитися залежно від складності програми та способів її обробки керованими користувачами даними.

Ефективне запобігання вразливостям XSS передбачає поєднання наступних заходів:

- фільтрування вводу при прибутті. У місці, де отримується ввід користувача, фільтрування має бути налаштовано найбільш суворо, виходячи з очікуваного або дійсного введення;
- кодування даних на виході. У місці, де керовані користувачем дані виводяться у відповіді HTTP, слід кодувати вихід, щоб запобігти його інтерпретації як активного вмісту. Залежно від вихідного контексту для цього може знадобитися застосування комбінацій кодування HTML, URL, JavaScript та CSS;
- слід використовувати відповідні заголовки відповідей. Щоб запобігти XSS у відповідях HTTP, які не мають на меті містити HTML або JavaScript, можна скористатися заголовками Content-Type та X-Content-Type-Options, щоб переконатися, що браузери інтерпретують відповіді так, як було задумано;
- політика безпеки вмісту. В якості додаткової міри захисту, можливо використовувати політику безпеки вмісту, щоб зменшити серйозність будь -яких вразливостей XSS, які все ще виникають.

Наведемо приклад сценарію даного типу атак.

Додаток використовує неперевірені дані при створенні HTML сніпету без їх підтвердження або перетворення:

(String) page + = "<input name = 'creditcard' type = 'TEXT'

value = '"+ request.getParameter (" CC ") +"'> ";

Зловмисник змінює параметр 'CC' в браузері на:

'> <Script> document.location = 'Http://www.attacker.com/cgi-bin/cookie.cgi?

foo = '+ document.cookie </ script>'.

Ідентифікатор сесії жертви відправляється на сайт зловмисника, дозволяючи атакуючому перехопити поточну сесію користувача.

Також,зловмисник може використовувати XSS для обходу захисту від межсайтової підміни запитів (CSRF), використовуваної в додатку.

3.8 Небезпечна десеріалізація

Серіалізація [13] - це процес перетворення складних структур даних, таких як об'єкти та їх поля, у "більш плоский" формат, який можна надсилати та приймати як послідовний потік байтів. Серіалізація даних значно спрощує:

- запис складних даних в міжпроцесову пам’ять, файл або базу даних;
- відправку складних даних, наприклад, по мережі, між різними компонентами програми або під час виклику API.

Важливо, що при серіалізації об’єкта його стан також зберігається. Іншими словами, атрибути об’єкта зберігаються разом із призначеними значеннями.

Десеріалізація - це процес відновлення цього байтового потоку до повністю функціональної репліки вихідного об’єкта в тому самому стані, як і під час його серіалізації. Тоді логіка веб-сайту може взаємодіяти з цим десеріалізованим об’єктом, так само як і з будь-яким іншим об’єктом.

Багато мов програмування пропонують рідну підтримку серіалізації. Те, як об’єкти серіалізуються, залежить від мови. Деякі мови серіалізують об’єкти у двійкових форматах, тоді як інші використовують різні формати рядків із різним ступенем прозорості для людей. При цьому всі атрибути вихідного об’єкта зберігаються у серіалізованому потоці даних, включаючи будь-які приватні поля. Щоб запобігти серіалізації поля, його слід явно позначити як "перехідний" у оголошенні класу.

Небезпечна десеріалізація - це процес, при якому веб-сайт десеріалізує керовані користувачами дані. Це потенційно дозволяє зловмиснику маніпулювати серіалізованими об'єктами, щоб передавати шкідливі дані в код програми.

Можна навіть замінити серіалізований об'єкт на об'єкт зовсім іншого класу. Об’єкти будь-якого класу, доступні для веб-сайту, будуть десеріалізовані та створені екземплярами, незалежно від того, який клас очікується. З цієї причини небезпечна десеріалізація іноді відома як вразливість "ін'єкції об'єкта".

Об’єкт несподіваного класу може спричинити виняток. Однак до цього часу шкода вже може бути заподіяна. Багато атак на основі десеріалізації завершуються до завершення десеріалізації. Це означає, що сам процес десеріалізації може ініціювати атаку, навіть якщо власна функціональність веб - сайту безпосередньо не взаємодіє з шкідливим об’єктом. З цієї причини веб - сайти, логіка яких ґрунтується на строго набраних мовах, також можуть бути вразливими до цих методів.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : експлуатувати десеріалізацію складно, оскільки готові експлойти рідко можна використовувати без їх зміни або доопрацювання;

2) недоліки безпеки : деякі інструменти можуть виявляти помилки десеріалізації, але для їх підтвердження зазвичай потрібна участь фахівця. Очікується, що в міру розробки нових інструментів виявлення та усунення помилок десеріалізації даних про їх поширеність побільшає;

3) наслідки: наслідки помилок десеріалізації не можна недооцінювати. подібні помилки можуть призвести до віддаленого виконання коду. Наслідки для бізнесу залежать від критичності захисту програми і даних.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.8.

Таблиця 3.8 – Зведена інформація про загрози небезпечної десеріалізації

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Небезпечна десеріалізація	Залежить від додатку	Складно: 1	Поширена : 2	Середньо: 2	Важкі: 3	15

Додатки та API уразливі, якщо здійснюють десеріалізацію шкідливих або модифікованих об'єктів, що надаються зловмисником. Це дозволяє здійснити два основних типу атак:

- атаки, пов'язані зі структурою об'єктів і даних, коли зловмисник змінює логіку додатка або віддалено виконує довільний код при наявності доступних з додатку класів, поведінка яких може змінюватися під час або після десеріалізації;
- атаки з підміною даних, наприклад, пов'язані з управлінням доступом, коли використовуються існуючі структури даних, але змінюється вміст.

Серіалізація може використовуватися в додатках для:

- віддаленої і міжпроцесесної взаємодії (RPC / IPC);
- дротових протоколів, веб-служб, брокерів повідомлень;
- кешування або збереження даних;
- баз даних, серверів кешування, файлових систем;
- cookie файлів HTTP, параметрів HTML-форм, токенів аутентифікації API.

Небезпечна десеріалізація зазвичай виникає через загальне нерозуміння того, наскільки небезпечною може бути десеріалізація керованих користувачами даних. В ідеалі, введення користувача ніколи не повинно взагалі десеріалізуватися.

Однак іноді власники веб-сайтів вважають себе захищеними, оскільки впроваджують певну форму додаткової перевірки десеріалізованих даних. Цей підхід часто є неефективним, оскільки практично неможливо здійснити перевірку чи санітарну обробку, щоб врахувати кожну можливість. Ці перевірки також є принципово хибними, оскільки вони залежать від перевірки даних після їх десеріалізації, що у багатьох випадках буде запізно, щоб запобігти атаці.

Вразливі місця також можуть виникнути через те, що десеріалізовані об’єкти часто вважаються надійними. Особливо при використанні мов з двійковим форматом серіалізації розробники можуть подумати, що користувачі не можуть ефективно читати або обробляти дані. Однак, хоча це може вимагати

додаткових зусиль, зловмисник може так само використовувати двійкові серіалізовані об'єкти, як і формати на основі рядків.

Атаки на основі десеріалізації також стали можливими через кількість залежностей, які існують на сучасних веб-сайтах. Типовий сайт може реалізовувати багато різних бібліотек, кожна з яких також має свої залежності. Це створює величезну групу класів і методів, якими важко безпечно керувати. Оскільки зловмисник може створювати екземпляри будь -якого з цих класів, важко передбачити, які методи можна викликати для шкідливих даних. Це особливо вірно, якщо зловмисник може з'єднати разом довгу серію несподіваних викликів методів, передаючи дані в стовбур, повністю не пов'язаний з первинним джерелом. Тому передбачити потік шкідливих даних і забити кожну потенційну діру практично неможливо.

Ефективне запобігання вразливостям десеріалізації передбачає поєднання наступних заходів:

- слід уникати десеріалізації введення даних користувачами, якщо це не є крайньою необхідністю. Високий ступінь потенційних можливостей та труднощі захисту від них у багатьох випадках переважують переваги;
- якщо потрібно десеріалізувати дані з ненадійних джерел, слід включити надійні заходи захисту, щоб переконатися, що дані не підроблені. Наприклад, можливо реалізувати цифровий підпис для перевірки цілісності даних. Однак слід пам’ятати, що перед початком процесу десеріалізації необхідно здійснити перевірки. В іншому випадку вони не приносять користі;
- якщо можливо, слід уникати використання загальних функцій десеріалізації. Серійні дані з цих методів містять усі атрибути вихідного об’єкта, включаючи приватні поля, які потенційно містять конфіденційну інформацію. Замість цього, можливо створити власні методи серіалізації, специфічні для класів, щоб було можливо принаймні контролювати, які поля відображаються;

– слід пам’ятати, що вразливістю є десеріалізація введення даних користувача, а не наявність ланцюжків гаджетів, які згодом обробляють дані. Не треба покладатися на спроби усунути ланцюжки гаджетів, які ідентифікуєте під час тестування. Недоцільно намагатися підключити їх усі через мережу міжбібліотечних залежностей, які майже напевно існують на веб-сайті. У будь -який момент, публічно задокументовані експлойти корупції пам'яті також є чинником, що означає, що програма може бути вразливою незалежно від цього.

Наведемо приклади сценаріїв даного типу атак.

1) React-додаток викликає набір мікрослужб Spring Boot. Будучи функціональними програмістами, розробники спробували забезпечити незмінність свого коду. Їх рішення полягає в серіалізації стану користувача і передачі його з кожним запитом. Зловмисник, помітивши підпис Java об'єкта "rO0", може використовувати Java Serial Killer для віддаленого виконання коду на сервері додатку.

2) На PHP форумі використовується серіалізація PHP об'єктів для зберігання "super-cookie", що містять ідентифікатор, роль, хеш пароля та інші дані користувача:

a: 4: {i: 0; i: 132; i: 1; s: 7: "Mallory"; i: 2; s: 4: "user";
i: 3; s: 32: "b6a8b3bea87fe0e05022f8f3c88bc960";}

Зловмисник змінює серіалізований об'єкт, наділяючи себе привілеями адміністратора:

a: 4: {i: 0; i: 1; i: 1; s: 5: "Alice"; i: 2; s: 5: "admin";
i: 3; s: 32: "b6a8b3bea87fe0e05022f8f3c88bc960";}

3.9 Використання компонентів з відомими вразливостями

Відомі вразливості [14] - це вразливості, виявлені в компонентах з відкритим кодом та опубліковані в спеціалізованій літературі, рекомендаціях з безпеки або засобах відстеження проблем. З моменту публікації вразливість може бути використана хакерами, які знаходять документацію. За статистичними даними, проблема використання компонентів з відомими вразливими місцями є надзвичайно поширеною.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : незважаючи на простоту пошуку вже готових експлойтів для більшості відомих вразливостей, деякі з них вимагають створення спеціальних засобів для їх експлуатації ;

2) недоліки безпеки : дана уразливість є дуже поширеною. Шаблони для розробників, що містять велику кількість компонентів, можуть призвести до непорозумінь того, які компоненти реально використовуються в додатку або API. Деякі сканери, такі як retire.js, можуть допомогти з виявленням вразливостей, але визначення складності їх експлуатації потребує додаткових зусиль;

3) наслідки : незважаючи на те, що не всі уразливості призводять до серйозних наслідків, причиною деяких масштабних зломів стали саме компоненти, що містять відомі уразливості. Залежно від активів, що підлягають захисту, подібна загроза може виявитися на вершині вашого списку пріоритетів.

За наведеними даними та розробленою методологію можна скласти зведену інформацію про даний тип погрози. Данна інформація наведена у таблиці 3.9.

Таблиця 3.9 – Зведена інформація про загрози використання компонентів з відомими вразливостями

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Використання компонентів з відомими вразливостями	Залежить від додатку	Середньо: 2	Дуже поширена : 3	Середньо: 2	Помірні: 2	14

Визначити, що наш додаток вразливий до даної погрози можливо якщо:

- ви не знаєте версії всіх використовуваних (на стороні клієнта і на стороні сервера) компонентів. Сюди відносяться самі компоненти і вбудовані залежності;
- ПО містить уразливості, не підтримується або застаріло. Сюди відносяться ОС, веб-сервери, сервери додатків, бази даних, додатки, API, а також всі компоненти, середовища виконання і бібліотеки;
- пошук вразливостей виконується нерегулярно;
- своєчасно не встановлюються виправлення або оновлення для використовуваних платформ, фреймворків і залежностей. Зазвичай таке відбувається, коли наявність оновлень перевіряється раз на місяць або квартал, в результаті чого організації тижнями або місяцями не усувають виправлені вразливості;
- розробники ПЗ не тестують сумісність оновлених або виправлених бібліотек;
- не забезпечується безпека компонентів

Компоненти зазвичай запускаються з привілеями додатку, тому вразливість в будь-якому з компонентів може привести до серйозних наслідків. Вразливість може з'явитися випадково (наприклад, через помилку в коді) або навмисно (наприклад, бекдор). Наводимо приклади експлуатації вразливостей, виявлених в компонентах:

- CVE-2017-5638: вразливість в Struts 2, що дозволяє віддалено виконати довільний код на сервері, стала причиною декількох серйозних зломів;
- вразливості в інтернеті речей (IoT) часто складно або неможливо усунути, а це може привести до серйозних наслідків (наприклад, в разі біомедичних приладів).

Існують автоматизовані інструменти, що дозволяють зловмисникам знаходити уразливі або некоректно налагоджені системи. Наприклад, пошукова система Shodan для IoT дозволяє виявити пристрої, в яких до сих пір не усунена уразливість Heartbleed, яка була виправлена в квітні 2014 року.

3.10 Недоліки журналювання і моніторингу

Журналювання [15] - це термін, що означає управління журналами.

Журнали - це записи подій, де збираються події, пов'язані зі станом системи.

Існує безліч журналів для різних систем. Наприклад, розглянемо веб-додаток: журнали можуть бути будь-якими діями, що виконуються у веб-службі, такими як підключення користувача до платформи, генерація помилок HTTP або доступ до ресурсу на сервері.

Щоб журнали були корисними, вони вимагають наступних дій:

- вибір корисної інформації для зберігання та архівування;
- забезпечення безпеки та конфіденційності збережених журналів;
- контроль якості даних журналу шляхом аналізу та додавання до журналів відсутньої інформації;
- аналіз журналів;
- контекстуалізація подій (збагачення журналу) за допомогою додаткових даних.

Контекстуалізація журналів - це функція, що вимагає найбільшого досвіду та знань системи контролю, щоб знати, яку інформацію слід зберігати, а яку ні.

Налаштовані журнали дозволяють досліджувати несправність у програмі, щоб запобігти її повторенню. У разі атаки журналювання дає змогу дізнатися виконавців, що стали причиною інциденту. Крім того, можливо дізнатися, які функції були використані, щоб виправити недолік, який дозволив атаку.

Моніторинг - це можливість знати загальний стан додатку в певний момент, а також історію минулих станів за кількома елементами:

- продуктивність - час відгуку різних ресурсів сервера;
- цілісність - перевірка того, що вміст веб -сторінок не змінюється;
- доступність - перевірка того, що програма повністю функціональна.

Моніторинг також важливий для виявлення будь-якої недостачі в продуктивності сервера та виявлення атак у режимі реального часу.

Для виявлення інцидентів безпеки, глобальний інструмент нагляду використовується для централізації різних журналів. Цей інструмент потребує опитування в режимі реального часу служб, які підлягають моніторингу. Він може базуватися на кількох елементах, званих метриками, такими як:

- навантаження процесора;
- кількість одночасних з'єднань;
- помилки сервера;
- моделювання взаємодії з додатком;
- навантаження мережі (QOS, затримка, пінг);
- спроби підключення, заблоковані брандмауером (виявлення Nmap).

Нагляд за цими елементами повинен дозволяти створювати події. Події є значними змінами стану, це можуть бути занадто велике навантаження на процесор, поштовх до сховища, помилка збірки, занадто велика кількість одночасних TCP -з'єднань. Для ефективного подальшого спостереження необхідно встановити рівні критичності подій. Це дозволяє обробляти їх у порядку пріоритетності, як у додатку для управління квитками.

Журналювання та моніторинг часто вважаються однаковими, оскільки система моніторингу містить журнали як основні дані, а без журналів якості

відсутній ефективний моніторинг. Однак аналіз журналу не слід плутати з моніторингом. Аналіз журналу - це робота після інциденту, тоді як моніторинг- це постійна робота.

Реалізація даних функцій може бути дуже складною. Для успішної реалізації потрібно вміти зберігати, сортувати та обробляти весь обсяг інформації. Без належного знання елементів, що підлягають моніторингу, може виникнути кілька проблем:

- незареєстрована зміна стану;
- запис тільки системних помилок, що не дозволяє впоратися з усіма проблемами; запис занадто багатьох елементів, швидко може виникнути проблема з місцем для зберігання;
- забагато інформації для пошуку;
- відсутність кореляції між даними;
- неправильна конфігурація подій.

Накопичення цих проблем робить журнали непридатними для використання. Тоді системи моніторингу стають скоріше обмеженням та втратою часу, ніж допомогою. Дана ситуація відома як недоліки ведення журналу та моніторингу, що може швидко стати великою проблемою та важливою вразливістю.

Аналізуючи даний тип загроз, можна представити інформацію про нього за наступними категоріями:

1) вектори атаки : експлуатація недоліків журналювання і моніторингу лежить в основі майже всіх великих зломів. При проведенні атак зловмисники покладаються на відсутність контролю і своєчасного реагування на інциденти;

2) недоліки безпеки : одним із способів визначити якість моніторингу є аналіз журналів після проведення тесту на проникнення. Для визначення можливої шкоди всі дії тестувальників повинні реєструватися відповідним чином;

3) наслідки : більшість атак починаються з аналізу вразливостей. Можливість проведення подібного аналізу підвищує ймовірність вдалою

експлуатації уразливості практично до 100%. У 2016 році виявлення факту проникнення займало в середньому 191 день - нанесений за цей час збитки міг бути величезним.

Таблиця 3.10 – Зведена інформація про загрози недоліків журналювання і моніторингу

Загроза	Джерело загрози	Складність експлуатації	Поширеність	Складність виявлення	Технічні наслідки	Сукупний ризик
Недоліки журналювання і моніторингу	Залежить від додатку	Середньо: 2	Дуже поширена : 3	Складно: 1	Помірні: 2	12

Недоліки журналювання, виявлення атак, моніторингу та реагування на інциденти виявляються постійно у наступних випадках:

- піддаються аудиту події, такі як вдалі і невдалі спроби входу в систему, а також важливі транзакції, які не реєструються;
- попередження і помилки не реєструються або реєструються некоректно;
- журнали додатків і API не перевіряються на предмет підозрілої активності;
- журнали зберігаються тільки локально;
- порогові значення попереджень і схеми реагування на інциденти відсутні або є неефективними;
- тестування на проникнення і сканування інструментами DAST не видають попереджень;
- додаток не може визначати, реагувати або попереджати про атаки в реальному або майже реальному часі.

Наведемо найкращі практики, які слід запровадити для полегшення впровадження та підвищення ефективності систем журналювання та моніторингу :

- обрати необхідні показники: розмежувати ті показники, що слід реєструвати, та ті що не слід ;
- визначити поточні показники: ті показники, що користувач повинен зареєструвати. Для цього спочатку рекомендується класифікувати показники, від найменш важливих (інформація, що не є корисною для забезпечення захисту вашої програми), до найважливіших (критична інформація);
- визначити свої події. Події, викликані системою моніторингу, мають стосуватися лише тих показників, які були щойно обрані. Також слід визначити для кожного показника, з якого порогу буде викликана подія;
- класифікувати свої події (інформаційні, низькі, середні, важливі, критичні): залежно від їх класифікації можуть бути активовані певні дії;
- забезпечити ітеративність керування та класифікація показників. Слід повертатися до обраних правил щоразу, коли впроваджуєте функції. Крім того, якщо елемент регулярно викликає попередження, яке не є корисним (хибно-позитивне), класифікацію необхідно переглянути;
- забезпечити окремі журнали. Журнали веб-додатку повинні містити лише інформацію, що стосується функціональних можливостей програми, а не проблем, властивих серверу;
- визначити структуру журналу. Журнали повинні мати формат, що дозволяє легко відфільтрувати корисну інформацію;
- централізувати журнали. Контроль повинен бути централізованим і керуватися зовнішнім сервером програми. Це дозволяє системі отримувати всю інформацію, щоб додати контекст до неї та співвідносити її;
- забезпечити наявність дублювання нагляду (резервного копіювання);
- обрати фреймворк ведення журналу, що відповідає інфраструктурі програми;
- документувати інфраструктуру, яка керує журналами;
- проводити оцінку системи реєстрації та моніторингу.

Наведемо приклади сценаріїв даного типу атак.

1) Форум відкритого проекту, який використовується невеликою командою, був зламаний через вразливість в його ПО. Зловмисники видалили внутрішній репозиторій, що містив наступну версію продукту, а також весь вміст форуму. Незважаючи на можливість відновлення джерела, відсутність моніторингу, журналирования або сповіщень призвело до більш серйозних наслідків. Через інцидент програмний проект з форуму більш не розвивається.

2) Зловмисник може використовувати один стандартний пароль для перевірки доступу до всіх облікових записів. До деяких з них він може підійти. Для інших буде зареєстрована лише невдала спроба входу. Через кілька днів спроба може повторитися, але вже з іншим паролем.

3) У великої торгівельної мережі є пісочниця для внутрішнього аналізу шкідливих вкладень. засоби пісочниці виявили потенційно шкідливе ПО, але ніхто не звертав уваги на одержані від пісочниці попередження, поки злом не виявили в зв'язку з шахрайськими транзакціями по банківськими картками від стороннього банку.

4 ОПИС ЗАХИСТУ ВІД ЗАГРОЗ ЗА НД ТЗІ 2.5-010-03

Рекомендації по захисту веб додатків описані в НД ТЗІ 2.5-010-03 [16]. Даний документ описує різні норми щодо захисту від загрозу. Наведемо деякі з них:

1) Установа, під час створення WEB-сторінки та визначення операторів, вузли яких будуть використовуватися для підключення до мережі Інтернет, повинна керуватися законами України, іншими нормативно-правовими актами, що встановлюють вимоги з технічного захисту інформації.

2) WEB-сторінка установи може бути розміщена на власному сервері або на сервері, що є власністю оператора. Власник сервера зобов'язаний гарантувати власнику інформації рівень захисту у відповідності до вимог НД ТЗІ 2.5-010-03

3) Функціонування WEB-сторінки забезпечується АС, за допомогою якої здійснюється актуалізація розміщених на WEB-сторінці інформаційних ресурсів та керування доступом до них. Для забезпечення захисту інформації WEB-сторінки в цій АС створюється КСЗІ, що є сукупністю організаційних і інженерно-технічних заходів, а також програмно-апаратних засобів, які забезпечують захист інформації.

4) Перелік інформації, призначеної для публічного розміщення на WEB-сторінці, визначається з урахуванням вимог діючого законодавства та затверджується керівником установи, що є власником WEB-сторінки.

5) Організація робіт із захисту інформації та забезпечення контролю за станом її захищеності на WEB-сторінці в установі здійснюється відповідальним підрозділом або відповідальною особою (далі - службою захисту інформації, СЗІ).

6) Захист інформації на всіх етапах створення та експлуатації WEB-сторінки здійснюється відповідно до розробленого установою плану захисту інформації, зміст якого визначено НД ТЗІ 1.4-001. План захисту затверджується

керівником установи, а у випадку використання сервера оператора – погоджується з власником сервера.

До складу АС, яка забезпечує функціонування WEB-сторінки, входять: ОС, фізичне середовище, в якому вона знаходиться і функціонує, середовище користувачів, оброблювана інформація, у тому числі й технологія її оброблення. Під час забезпечення захисту інформації мають бути враховані всі характеристики зазначених складових частин, які впливають на реалізацію політики безпеки WEB-сторінки.

Політика безпеки інформації в АС повинна поширюватися на об’єкти комп’ютерної системи, які безпосередньо чи опосередковано впливають на безпеку інформації.

До таких об’єктів належать:

- адміністратор безпеки та співробітники СЗІ;

- користувачі, яким надано повноваження забезпечувати управління АС;

- користувачі, яким надано право доступу до загальнодоступної інформації;

- інформаційні об’єкти, що містять загальнодоступну інформацію;

- системне та функціональне ПЗ, яке використовується в АС для оброблення інформації або для забезпечення функцій КЗЗ;

- технологічна інформація КСЗІ (дані про мережеві адреси, імена, персональні ідентифікатори та паролі користувачів, їхні повноваження та права доступу до об’єктів, встановлені робочі параметри окремих механізмів або засобів захисту, інша інформація баз даних захисту, інформація журналів реєстрації дій користувачів тощо);

- засоби адміністрування і управління обчислювальною системою АС та технологічна інформація, яка при цьому використовується;

- обчислювальні ресурси АС (наприклад, дисковий простір, тривалість сеансу роботи користувача із засобами АС, час використання центрального процесора і т. ін.), безконтрольне використання або захоплення яких окремим

користувачем може призвести до блокування роботи інших користувачів, компонентів АС або АС в цілому.

З урахуванням особливостей надання доступу до інформації WEB-сторінки, типових характеристик середовищ функціонування та особливостей технологічних процесів оброблення інформації у додатку, в НД ТЗІ НД ТЗІ 2.5-010-03 визначаються наступні мінімально необхідні рівні послуг безпеки для забезпечення захисту інформації від загроз:

- за умови, коли WEB-сервер і робочі станції розміщуються на території установи-власника WEB-сторінки або на території оператора (технологія Т1), мінімально необхідний функціональний профіль визначається:

КА-2, ЦА-1, ЦО-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1;

- за умови, коли WEB-сервер розміщується у оператора, а робочі станції – на території власника WEB-сторінки, взаємодія яких з WEB-сервером здійснюється з використанням мереж передачі даних (технологія Т2), мінімально необхідний функціональний профіль визначається:

КА-2, КВ-1, ЦА-1, ЦО-1, ЦВ-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1,НЦ-1, НТ-1, НВ-1.

Технологія Т1 різниться від технології Т2 способом передачі інформації від робочої станції до WEB-сервера, а саме: наявністю у другому випадку незахищеного середовища, яке не контролюється, і додатковими вимогами щодо ідентифікації та автентифікації між КЗЗ робочої станції й КЗЗ WEB-сервера під час спроби розпочати обмін інформацією та забезпечення цілісності інформації при обміні.

За власником WEB-сторінки залишається право реалізації, у разі необхідності, окремих послуг безпеки інформації зазначених профілів з більш високим рівнем, доповнення цих профілів іншими послугами, а також реалізація послуг безпеки з більш високим рівнем гарантій.

Також для забезпечення безпеки за НД ТЗІ 2.5-010-03 повинні бути реалізовані функціональні послуги безпеки інформації. Перелік необхідних для реалізації послуг наведно в таблиці 4.1.

Таблиця 4.1 – Функціональні послуги безпеки інформації

№	Функціональна послуга безпеки інформації	Вимоги до реалізації
1	Базова адміністративна конфіденційність	КЗЗ повинен реалізувати рівень КА-2. Ця послуга дозволяє адміністратору безпеки керувати потоками інформації від захищених об'єктів до користувачів.
2	Конфіденційність при обміні	КЗЗ повинен реалізувати рівень КВ-1. Ця послуга дозволяє забезпечити захист об’єктів від несанкціонованого ознайомлення з інформацією, що міститься в них, під час їх експорту/імпорту через незахищене середовище.
3	Мінімальна адміністративна цілісність	КЗЗ повинен реалізувати рівень ЦА-1. Ця послуга дозволяє керувати потоками інформації від користувачів до захищених об’єктів WEB-сторінки
4	Цілісність при обміні	КЗЗ повинен реалізувати рівень ЦВ-1. Ця послуга дозволяє забезпечити захист WEB-сторінки від несанкціонованої модифікації інформації, яка передається між WEB-сервером та робочими станціями у разі використання технології Т2, під час експорту/імпорту інформації через незахищене середовище. Політика послуги стосується всіх об’єктів, що передаються.
5	Відкат	КЗЗ повинен реалізувати рівень ЦО-1. Ця послуга забезпечує можливість відмінити окрему операцію або послідовність операцій і повернути захищений об'єкт після внесення до нього змін до попереднього наперед визначеного стану.

Продовження таблиці 4.1

6	Використання ресурсів	КЗЗ повинен реалізувати рівень ДР-1. Ця послуга дозволяє керувати використанням користувачами послуг та ресурсів.
7	Відновлення після збоїв	КЗЗ повинен реалізувати рівень ДВ-1. Політика відновлення після збоїв, що реалізується КЗЗ, стосується: системного та функціонального програмного забезпечення; засобів захисту інформації та засобів управління КСЗІ; засобів адміністрування та управління обчислювальною системою АС – і гарантує повернення АС у відомий захищений стан після відмов або переривання обслуговування, спричинених помилковими діями користувачів, неврахованою функціональною недостатністю програмного та апаратного забезпечення
8	Реєстрація	КЗЗ повинен реалізувати рівень НР-2. Послуга дозволяє контролювати небезпечні відповідно до політики безпеки WEB-сторінки дії користувачів всіх категорій із захищеними об’єктами.
9	Ідентифікація і автентифікація	КЗЗ повинен реалізувати рівень НИ-2. Ідентифікація і автентифікація дозволяють КЗЗ визначити і перевірити особу суб’єкта, що намагається одержати доступ до захищених об’єктів WEB-сторінки.

Продовження таблиці 4.1

10	Ідентифікація і автентифікація при обміні	КЗЗ повинен реалізувати рівень НВ-1. Ця послуга дозволяє у разі використання технології Т2 компонентам КЗЗ WEB-сервера і віддаленої робочої станції здійснити взаємну ідентифікацію, перш ніж розпочати взаємодію.
11	Достовірний канал	КЗЗ повинен реалізувати рівень НК-1. Ця послуга повинна гарантувати користувачу будь-якої категорії можливість безпосередньої взаємодії з КЗЗ, а також те, що ніяка взаємодія користувача з АС не може бути модифікованою іншим користувачем або процесом. Послуга визначає вимоги до механізму встановлення достовірного зв'язку між користувачем і КЗЗ.
12	Розподіл обов’язків	КЗЗ повинен реалізувати рівень НО-1. Ця послуга дозволяє розмежувати повноваження користувачів, визначивши категорії користувачів з певними і притаманними для кожної з категорій функціями (ролі). Послуга призначена для зменшення потенційних збитків від навмисних або помилкових дій користувачів і обмеження авторитарності керування АС.
13	Цілісність комплексу засобів захисту	КЗЗ повинен реалізувати рівень НЦ-1. Ця послуга визначає міру здатності КЗЗ WEB-сторінки захищати себе і гарантувати свою спроможність керувати захищеними об’єктами. Політика цілісності КЗЗ повинна визначати склад КЗЗ, механізми контролю цілісності його компонентів та порядок їх використання.

Продовження таблиці 4.1

14	Самотестування	КЗЗ повинен реалізувати рівень НТ-1. Самотестування дозволяє КЗЗ перевірити і на підставі цього гарантувати правильність функціонування і цілісність певної множини функцій захисту WEB-сторінки. Політика самотестування поширюється на адміністратора безпеки, компоненти системного та функціонального програмного забезпечення, які задіяні для реалізації механізмів КЗЗ, засоби захисту інформації.

5 РЕКОМЕНДАЦІЇ КОМАНДІ ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ

Внаслідок детального аналізу переліку найнебезпечніших для веб додатків загроз , шляхів їх реалізації та державних стандартів з забезпечення безпеки, стає можливим розробити комплекс рекомендацій для всієї команди проекту по захисту веб додатку від погроз.

Сформовані рекомендації до розробників наводимо у таблиці 5.1

Таблиця 5.1 – Рекомендації розробникам для забезпечення безпеки

№	Компонент	Рекомендація
1	Вимоги до безпеки додатку	Щоб створити безпечне веб-додаток, необхідно спочатку розробити план захисту інформації відповідно до НД ТЗІ 1.4-001
2	Архітектура безпеки додатку	Замість додавання механізмів забезпечення безпеки в готові програми та API економічно вигідніше вбудовувати їх на етапі розробки. В якості керівництва при розробці безпечного додатку з нуля рекомендується використовувати технічне завдання, розроблене згідно з НД ТЗІ 3.7-001.
3	Стандартні методи забезпечення безпеки	Складно створити надійні і практичні засоби забезпечення безпеки. Використання стандартних засобів значно спрощує розробку безпечних додатків і API.

Сформовані рекомендації до тестувальників наводимо у таблиці 5.2

Таблиця 5.2 – Рекомендації тестувальникам для забезпечення безпеки

№	Компонент	Рекомендація
1	Розуміння моделі загроз	Перед початком тестування переконайтеся, що правильно розставили пріоритети, виходячи з моделі загроз. Якщо у вас немає моделі, її необхідно розробити.
2	Стратегії тестування	Виберіть самий простий, швидкий і точний спосіб перевірки кожної вимоги. Для цього використовуйте розроблені за НД ТЗІ 2.5-010-03 функціональні і нефункціональні вимоги до безпеки та дані отримані після аналізу найпопулярніших загроз безпеці додатку.
4	Повідомляйте про результати правильно	Не важливо як багато недоліків у безпеці знайде тестувальник, якщо вони не будуть виправлені. Тому усі найдені помилки мають буди задокументовані як можливо детальніше . Рекомендується використовувати баг трекінгову систему JIRA

Сформовані рекомендації до менеджерів наводимо у таблиці 5.3

Таблиця 5.3 – Рекомендації менеджерам для забезпечення безпеки

№	Компонент	Рекомендація
1	Вимоги та управління ресурсами	Організуйтесь і обговоріть з замовником бізнес-вимоги до додатку, включаючи забезпечення конфіденційності, достовірності, цілісності та доступності всіх інформаційних активів, а також очікувану бізнес-логіку

Продовження таблиці 5.3

		Складіть перелік технічних вимог, включаючи функціональні і нефункціональні вимоги з безпеки.
		Сплануйте і обговоріть бюджет, що охоплює всі аспекти проектування, створення, тестування і експлуатації, а також роботи по забезпеченню безпеки.
2	Планування і проектування	Обговоріть плани і проекти з розробниками і внутрішніми партнерами, наприклад, фахівцями з безпеки.
		Визначте архітектуру і засоби управління безпекою, а також контрзаходи, відповідні вимогам захисту і очікуваним рівнями небезпеки. Все це повинно забезпечуватися фахівцями з безпеки.
		Переконайтеся, що власник додатку приймає інші ризики або надає додаткові ресурси.
		У кожному спринті забезпечте створення записів з безпеки із зазначенням обмежень, доданих для нефункціональлних вимог.
3	Розгортання, тестування і впровадження	Автоматизуйте безпечне розгортання додатку, інтерфейсів і компонентів, а також отримання необхідних дозволів.
		Протестуйте технічні можливості і інтеграцію з IT-архітектурою, а також організуйте бізнес тестування.
		Протестуйте "штатне" і "нештатне" використання технічних і виробничих можливостей.
		Організуйте тестування безпеки відповідно до внутрішніх процесів, вимог захисту і передбаченим рівнем небезпеки для кожної програми.

Продовження таблиці 5.3

		Введіть додаток в експлуатацію і перестаньте використовувати старі додатки. При необхідності узгодьте всю документацію, а також базу даних контролю змін і архітектуру безпеки
4	Інтеграція безпеки в існуючі процеси	Визначте і упровадите в існуючі процеси розробки та експлуатації заходи щодо безпечної реалізації і контролю. Склад робіт: моделювання загроз, безпечне проектування і аналіз проектів, написання безпечного коду і його аналіз, тестування і усунення недоліків.
		Для досягнення успіху забезпечте наявність експертів в предметній області і служб підтримки для розробників і проектної команди.

При успішній та повної реалізації даних рекомендацій при розробці додатку, ризик для проектної команди розробити уразливий до проаналізованих загроз додаток та спричинити збитки замовнику через не усунені загрози безпеці додатку перестає існувати , так как розроблений додаток буде не вразливим до усіх проаналізованих найпопулярніших загроз безпеці на даний час.

6 РЕАЛІЗАЦІЯ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ ДОДАТКУ

При реалізації описаних рекомендацій, додаток має стати більш стійким до найпоширеніших загроз, але остаточне рішення про його безпечність та готовність ,в першу чергу , виносять тестувальники . Вони проводять детальний аналіз додатку та сповіщають менеджмент про всі виявлені вразливості.

При проведення тестування для забезпечення швидкості виконання та максимального покриття тестових випадків слід використовувати тест-план.

Тест-план - документ, що описує і регламентує перелік робіт по тестування, а також відповідні техніки і підходи, стратегію, області відповідальності, ресурси, розклад і ключові дати.

Кожен тест план має мету - гранично короткий опис мети розробки програми. Частково ця секція спирається на бізнес вимоги, але тут інформація подається в більш стислому вигляді з акцентом на найголовніших завданнях.

Далі обираються області, що піддаються тестуванню (перелік функцій або нефункціональних особливостей додатку, які будуть піддані тестуванню) та області, які не піддаються тестуванню (перелік функцій або не функціональних особливостей додатку, які не будуть піддані тестуванню). У контексті тест-плану націленого на тестування безпеки додатку верифікації підлягатимуть реалізації вимог з забезпечення безпеки.

Після цього розпочинається найважливіший розділ тест-плану – опис тестової стратегії. Тестова стратегія визначає процес тестування з точки зору застосовуваних методів, підходів, видів тестування, технологій, інструментальних засобів. Для розробки тестової стратегії, призначеної на тестування безпеки додатку, слід використовувати розроблені за НД ТЗІ 2.5-010-03 функціональні і нефункціональні вимоги до безпеки та дані отримані після аналізу найпопулярніших загроз безпеці додатку.

Після завершення опису тестової стратегії визначаються критерії тестування. До них належать : критерії початку тестування , критерії припинення

тестування , критерії відновлення тестування , критерії завершення тестування та критерії прийняття .

Також у тест плані повинен міститись перелік доступних для тестування ресурсів (програмних, апаратних, людських, фінансових та часових) та розклад проведення тестування, в якому буде зазначено, що і до якого моменту повинно бути зроблено.

Після розробки розкладу визначаються основні ролі та відповідальності при тестуванні , тобто приводиться перелік необхідних ролей і сфери відповідальності фахівців, що виконують ці ролі.

Важливу частину плану складає оцінка ризиків - перелік ризиків, які з високою ймовірністю можуть виникнути в процесі роботи над проектом. За кожним ризиком дається оцінка представляється їм загрози і наводяться варіанти виходу з ситуації.

Наприкінці тест плану визначається перелік необхідної документації та метрик. Документація складається з переліку використовуваної тестової документації з зазначенням, хто і коли повинен її готувати і кому передавати. Метрики ж являються числовими характеристиками показників якості додатку.

Спираючись на описані секції тест плану , функціональні і нефункціональні вимоги до безпеки та дані отримані після аналізу найпопулярніших загроз безпеці додатку розробимо шаблон тест-плану для тестування безпеки веб додатку . Проте слід мати на увазі, що процес тестування безпеки веб додатку у більшості випадків є складовою комплексного процесу тестування . У такому випадку пункти з розробленого тест плану для тестування безпеки веб додатку слід включати у загальний тест план.

Розроблений тест-план має ряд переваг для проекту в цілому :

- добре розпланований, систематизований підхід дозволяє досягти кращих результатів, а також дозволяє виявляти більшу кількість помилок, ніж неорганізована, погано розпланована діяльність;
- тест план дозволяє управляти процесом тестування більш ефективно;

- тест план дозволяє побачити і зрозуміти мінімальний рівень тестування і отримати уявлення про рівень проведеного тестування кожної області продукту;
- тест план дозволяє досягти угоди між виконавцями, замовником і менеджером, про те, яким чином і в які терміни буде проводитися тестування.

6.1 Шаблон тест плану

Мета : метою плану тестування є опис та формалізація процесу тестування безпеки веб -сайту.

Функціональні тести безпеки будуть виконані згідно з тестовою стратегією з метою виявлення вразливостей у безпеці веб сайту.

Не будуть виконуватися функціональні та не функціональні тести направлені на перевірку якості розробленого додатку.

Тестова стратегія : тестування буде проводитися для веб -сайту

Планується 3 етапи процесу тестування:

- перший етап - це аналіз специфікації, створення тест плану та частковий запуск функціональних тестів;
- другий етап буде присвячений детальному виконанню функціональних тестів з виявленням та описом дефектів;
- третій крок - перевірка вирішених дефектів та проведення регресійного тестування.

Основні види тестів безпеки веб сайту представлені у таблиці 6.1

Таблиця 6.1 – Основні тести з перевірки безпеки

№	Перевірка
1	Перевірте використання моделювання загроз для кожної зміни конструкції або спринтерського планування для виявлення загроз, планування контрзаходів, сприяння відповідним реагуванням на ризики та керівництва тестуванням безпеки.
2	Переконайтеся, що всі історії користувача та функції містять функціональні обмеження безпеки, наприклад "Як користувач я повинен мати можливість переглядати та редагувати свій профіль. Я не повинен мати змогу переглядати чи редагувати профіль будь-кого іншого"
3	Перевірте документацію та обґрунтування всіх меж довіри, компонентів та значних потоків даних програми.
4	Перевірте визначення та аналіз безпеки архітектури високого рівня програми та всіх підключених віддалених служб.
5	Перевірте впровадження централізованих, простих (економія дизайну), перевірених, безпечних та багаторазових засобів безпеки, щоб уникнути дублювання, відсутності, неефективності чи небезпеки контролю.
6	Перевірте наявність контрольного списку безпечного кодування, вимог безпеки, рекомендацій або політики для всіх розробників та тестувальників.
7	Перевірте використання унікальних або спеціальних облікових записів операційної системи з низькими привілеями для всіх компонентів програми, служб та серверів
8	Переконайтеся, що зв'язок між компонентами програми, включаючи API, проміжне програмне забезпечення та рівні даних, автентифікований. Компоненти повинні мати найменші необхідні привілеї.

Продовження таблиці 6.1

9	Переконайтеся, що програма використовує єдиний перевірений механізм автентифікації, який, як відомо, є безпечним, може бути розширений, включивши сильну автентифікацію, та має достатнє журналювання та моніторинг для виявлення зловживань або порушень облікового запису.
10	Переконайтеся, що всі шляхи автентифікації та API управління ідентифікацією реалізують послідовний рівень контролю безпеки автентифікації, щоб не було слабших альтернатив на ризик програми.
11	Переконайтеся, що обране рішення контролю доступу є достатньо гнучким для задоволення потреб програми.
12	Перевірте дотримання принципу найменших привілеїв у функціях, файлах даних, URL-адресах, контролерах, послугах та інших ресурсах. Це передбачає захист від підробки та підвищення привілеїв.
13	Переконайтеся, що програма використовує єдиний і перевірений механізм контролю доступу для доступу до захищених даних та ресурсів. Усі запити повинні проходити через цей єдиний механізм, щоб уникнути копіювання та вставлення або незахищених альтернативних шляхів.
14	Переконайтеся, що використовується атрибут або керування доступом на основі функцій, завдяки чому код перевіряє авторизацію користувача щодо елемента / елемента даних, а не просто їх ролі. Дозвіл все одно слід розподіляти за допомогою ролей.
15	Переконайтеся, що вимоги до введення та виведення чітко визначають, як обробляти та обробляти дані на основі типу, вмісту та чинного законодавства, нормативних актів та інших положень політики.

Продовження таблиці 6.1

16	Переконайтеся, що серіалізація не використовується під час спілкування з ненадійними клієнтами. Якщо це неможливо, переконайтеся, що застосовуються адекватні засоби контролю цілісності (і, можливо, шифрування, якщо надсилаються конфіденційні дані), щоб запобігти атакам десеріалізації, включаючи введення об'єкта.
17	Переконайтеся, що перевірка вхідних даних застосована на рівні надійного сервісу.
18	Переконайтеся, що вихідне кодування відбувається близько до інтерпретатора, для якого воно призначене.
19	Переконайтеся, що існує чітка політика щодо управління криптографічними ключами та що життєвий цикл криптографічного ключа відповідає такому стандарту управління ключами, як NIST SP 800-57.
20	Переконайтеся, що споживачі криптографічних послуг захищають ключові матеріали та інші секрети, використовуючи сховища ключів або альтернативи на основі API.
21	Переконайтеся, що всі ключі та паролі можна замінити та вони є частиною чітко визначеного процесу повторного шифрування конфіденційних даних.
22	Переконайтеся, що архітектура розглядає секрети на стороні клієнта, такі як симетричні ключі, паролі або маркери API, як небезпечні та ніколи не використовує їх для захисту або доступу до конфіденційних даних.
23	Переконайтеся, що у всій системі використовується загальний формат ведення журналу
24	Переконайтеся, що журнали надійно передаються в бажано віддалену систему для аналізу, виявлення, попередження та ескалації.

Продовження таблиці 6.1

25	Переконайтеся, що всі конфіденційні дані ідентифіковані та класифіковані за рівнями захисту.
26	Переконайтеся, що всі рівні захисту мають відповідний набір вимог захисту, таких як вимоги до шифрування, вимоги до цілісності, збереження, конфіденційності
27	Переконайтеся, що програма шифрує зв’язок між компонентами, особливо коли ці компоненти знаходяться в різних контейнерах, системах, сайтах або хмарних постачальниках.
28	Переконайтеся, що компоненти програми перевіряють справжність кожної сторони в лінії зв'язку, щоб запобігти атакам "людина посередині". Наприклад, компоненти програми повинні перевіряти сертифікати та ланцюжки TLS.
29	Переконайтеся, що завантажені користувачем файли зберігаються за межами веб-кореня.
30	Переконайтеся, що завантажені користувачем файли обслуговуються з не пов’язаного домену, наприклад, у хмарному сховищі файлів. Реалізуйте відповідну Політику безпеки вмісту (CSP), щоб зменшити ризик від векторів XSS або інших атак із завантаженого файлу.
31	Переконайтеся, що двійкові підписи, надійні з'єднання та перевірені кінцеві точки використовуються для розгортання двійкових файлів на віддалених пристроях.
32	Переконайтеся, що паролі, встановлені користувачами, мають принаймні 12 символів (після об’єднання кількох пробілів).
33	Переконайтеся, що паролі на більше 64 символів дозволені, але не більше 128 символів.
34	Перевірте, чи користувачі можуть змінити свій пароль.
35	Переконайтеся, що для зміни функцій пароля потрібен поточний і новий пароль користувача.

Продовження таблиці 6.1

36	Переконайтеся, що в наявності є вимірювач надійності пароля, щоб допомогти користувачам встановити надійніший пароль.
37	Переконайтеся, що паролі зберігаються у формі, стійкій до офлайн - атак.
38	Перевірте, чи немає підказок щодо пароля чи автентифікації на основі знань (так званих "секретних питань").
39	Перевірте, чи немає спільних облікових записів або облікових записів за умовчанням (наприклад, "root", "admin" або "sa").
40	Переконайтеся, чи затверджені криптографічні алгоритми використовуються для генерації та перевірки.
41	Перевірте, чи програма ніколи не розкриває маркери сеансу в параметрах URL.
42	Перевірте, чи програма генерує новий маркер сеансу для автентифікації користувача.
43	Переконайтеся, що маркери сеансу мають принаймні 64 біти ентропії.
44	Перевірте, чи програма зберігає в браузері лише маркери сеансу, використовуючи захищені методи, такі як належним чином захищені файли cookie
45	Переконайтеся, що маркер сеансу генерується за допомогою схвалених криптографічних алгоритмів
46	Переконайтеся, що програма дає можливість припинити всі інші активні сеанси після успішної зміни пароля
47	Переконайтеся, що у маркерів сеансів на основі файлів cookie встановлено атрибут «Secure»
48	Переконайтеся, що у маркерів сеансів на основі файлів cookie встановлено атрибут «HttpOnly»
49	Перевірте, що програма використовує маркери сеансу, а не статичні секрети та ключі API.

Продовження таблиці 6.1

50	Переконайтеся, що програма забезпечує повний, дійсний сеанс входу або вимагає повторної автентифікації або вторинної перевірки, перш ніж дозволити будь-які конфіденційні транзакції або зміни облікового запису.
51	Переконайтеся, що програма застосовує правила контролю доступу на довіреному рівні послуг, особливо якщо присутній контроль доступу на стороні клієнта і його можна обійти.
52	Переконайтеся, що кінцеві користувачі не можуть маніпулювати усіма атрибутами користувачів та даних та інформацією про політику, що використовуються засобами контролю доступу, якщо це не дозволено спеціально.
53	Перевірте, чи адміністративні інтерфейси використовують відповідну багатофакторну автентифікацію, щоб запобігти несанкціонованому використанню.
54	Переконайтеся, що програма має захист від атак забруднення параметрів HTTP, особливо якщо фреймворк програми не робить різниці щодо джерела параметрів запиту
55	Переконайтеся, що всі дані (поля форм HTML, запити REST, параметри URL, заголовки HTTP, файли cookie, пакетні файли, канали RSS тощо) перевірені за допомогою позитивної перевірки (списку дозволів).
56	Переконайтеся, що URL -адреси переспрямовують та пересилають лише адреси, які є у списку дозволених, або показують попередження під час переспрямування на потенційно ненадійний вміст.
57	Переконайтеся, що додаток захищає від атак із введенням шаблонів, переконавшись, що будь -який введений користувачем вхідний дані проходить санітарну обробку або є ізольованим.

Продовження таблиці 6.1

58	Переконайтеся, що бази даних (наприклад, SQL, HQL, ORM, NoSQL) використовують параметризовані запити, ORM, рамки сутностей захищені від атак ін’єкцій бази даних
59	Переконайтеся, що рядки формату не приймають потенційно вороже введення і є постійними.
60	Переконайтеся, що методи перевірки знаків, діапазонів та введення використовуються для запобігання переповненню цілих чисел.
61	Переконайтеся, що всі криптографічні модулі безпечно виходять з ладу, а помилки обробляються таким чином, що не дозволяє активувати атаки Padding Oracle.
62	Переконайтеся, що вектор ініціалізації шифрування, конфігурація шифру та режими блокування налаштовані надійно з використанням останніх порад.
63	Переконайтеся, що зашифровані дані автентифіковані за допомогою підписів, автентифікованих режимів шифрування або HMAC, щоб переконатися, що зашифрований текст не змінюється сторонніми сторонами.
64	Переконайтеся, що всі криптографічні операції є постійними, без операцій "короткого замикання" в порівняннях, обчисленнях або поверненнях, щоб уникнути витоку інформації.
65	Переконайтеся, що випадкові числа створюються з належною ентропією, навіть якщо додаток знаходиться під великим навантаженням, або що додаток витончено погіршується за таких обставин.
66	Переконайтеся, що рішення для управління секретами, наприклад сховище ключів, використовується для безпечного створення, зберігання, контролю доступу та знищення секретів.

Продовження таблиці 6.1

68	Переконайтеся, що програма не реєструє облікові дані або реквізити платежу. Маркери сеансів слід зберігати лише у журналах у незворотній хешованій формі.
69	Переконайтеся, що програма не реєструє інші конфіденційні дані, як це визначено місцевим законодавством про конфіденційність або відповідною політикою безпеки
70	Переконайтеся, що кожна подія журналу містить необхідну інформацію, яка дозволила б детально вивчити хронологію, коли подія відбувається.
71	Переконайтеся, що всі рішення щодо контролю доступу можуть реєструватися, а всі невдалі рішення реєструються. Це має включати запити з відповідними метаданими, необхідними для розслідування безпеки.
72	Перевірте, чи всі події захищені від ін’єкцій під час перегляду у програмному забезпеченні перегляду журналів.
73	Переконайтеся, що журнали безпеки захищені від несанкціонованого доступу та внесення змін.
74	Переконайтеся, що обробка винятків (або функціональний еквівалент) використовується у всій кодовій базі для обліку очікуваних та несподіваних умов помилок.
75	Перевірте, чи програма захищає конфіденційні дані від кешування в компонентах сервера, таких як балансири завантаження та кеші програм.
76	Перевірити, чи додаток мінімізує кількість параметрів у запиті, таких як приховані поля, змінні Ajax, файли cookie та значення заголовків.
77	Переконайтеся, що регулярно виконуються резервні копії важливих даних і чи виконується тестове відновлення даних.

Продовження таблиці 6.1

78	Переконайтеся, що резервні копії зберігаються надійно, щоб уникнути крадіжки чи пошкодження даних.
79	Переконайтеся, що програма встановлює достатньо заголовків проти кешування, щоб конфіденційні дані не кешувалися в сучасних браузерах.
80	Переконайтеся, що у користувачів є спосіб видалення або експорту даних за запитом.
81	Переконайтеся, що користувачам надано чітку формулювання щодо збору та використання наданої персональної інформації та вони надали згоду на використання цих даних, перш ніж вони якимось чином були використані.
82	Переконайтеся, що конфіденційна особиста інформація підлягає класифікації збереження даних, наприклад, що старі або застарілі дані видаляються автоматично, за розкладом або відповідно до ситуації.
83	Переконайтеся, що захищений TLS використовується для всіх підключень клієнта та не переходить до незахищених або незашифрованих протоколів.
84	Перевірте за допомогою онлайнових або найновіших інструментів тестування TLS, що ввімкнено лише надійні алгоритми, шифри та протоколи, а найсильніші алгоритми та шифри встановлені як бажані.
85	Переконайтесь, що помилки з'єднання бекенд TLS зареєстровані.
86	Переконайтеся, що використовується інструмент аналізу коду, який може виявити потенційно шкідливий код, такий як функції часу, небезпечні операції з файлами та мережеві з'єднання.
87	Переконайтеся, що програма не вимагає зайвих або надмірних дозволів на функції чи датчики, пов'язані з конфіденційністю, такі як контакти, камери, мікрофони чи місцезнаходження.

Продовження таблиці 6.1

88	Переконайтеся, що вихідний код програми та бібліотеки третіх сторін не містять потенційно небажаної функціональності.
89	Перевірте, що програма буде обробляти потоки бізнес -логіки для одного і того ж користувача в послідовному порядку кроків і без пропусків кроків.
90	Перевірте, що програма буде обробляти лише потоки бізнес -логіки, кроки котрої обробляються за реалістичний час для людини, тобто транзакції не надсилаються надто швидко.
91	Переконайтеся, що програма має налаштоване сповіщення, коли виявляються автоматичні атаки або незвична активність.
92	Переконайтеся, що програма не прийматиме великі файли, які можуть заповнити сховище або спричинити відмову в обслуговуванні.
93	Переконайтеся, що файли, отримані з ненадійних джерел, перевірені на очікуваний тип на основі вмісту файлу.
94	Переконайтеся, що метадані ненадійних файлів не використовуються безпосередньо з системним API або бібліотеками для захисту від введення команд ОС.
95	Переконайтеся, що прямі запити до завантажених файлів ніколи не виконуватимуться як вміст HTML/JavaScript.
96	Переконайтеся, що всі компоненти програми використовують однакові кодування та синтаксичні аналізатори, щоб уникнути атак синтаксичного аналізу, які використовують різні URI або поведінку файлу, які можуть використовуватися в атаках SSRF та RFI.
97	Переконайтеся, що доступ до функцій адміністрування та управління обмежений авторизованими адміністраторами.
98	Переконайтеся, що файли, отримані з ненадійних джерел, перевіряються антивірусними програмами, щоб запобігти завантаженню відомого шкідливого вмісту.

Продовження таблиці 6.1

99	Переконайтеся, що веб-сервер або сервер додатків налаштовано з дозволеним списком ресурсів або систем, до яких сервер може надсилати запити або завантажувати дані.
100	Переконайтеся, що стислі файли перевіряються на наявність "zip-бомб" - невеликих вхідних файлів, які розпаковуватимуться у величезні файли, таким чином вичерпуючи обмеження для зберігання файлів.

Критерії початку тестування: специфікація готова.

Критерії завершення тестування: усі тести проходяться успішно, основні помилки виправлені та перевірені, немає помилок з критичним пріоритетом та не більше 5 помилок з низьким пріоритетом.

У подальших секціях тест плану уся наведена інформація використовується лише в якості прикладу заповнення даних секцій.

Апаратні ресурси: ПК (16 ГБ оперативної пам’яті, i5 3.20 ГГц)

Ресурси програмного забезпечення: Google Chrome 76.0.3809.100 (x64) для Windows 10

Персонал:

Інженер з безпеки - Кочанов Максим.

Провідний інженер з безпеки - Олександр Федюшин.

Розклад робіт з тестування наведено у таблиці 6.2

Таблиця 6.2 – Розклад робіт з тестування

Дати	Тестова активність
21.10 -22.10	Аналіз специфікації
21.10 – 22.10	Створення тестів
23.10	Створення тест плану
24.10	Початок виконання тестів
29.10	Завершення виконання тестів
24.10-2.11	Пошук та опис дефектів
24.10-2.11	Перевірка виправлення дефектів
30.10	Початок повторного тестування
2.11	Завершення повторного тестування
3.11	Створення звіту з результатами тестування

Оцінка ризиків відбувається за показниками серйозності даного ризику (від 1 до 10) , імовірності його появлення (від 0 до 1) та його впливу на процес тестування (від 0 до 3).

Оцінка ризиків при виконанні робіт наведена у таблиці 6.3

Таблиця 6.3 – Оцінка ризиків

№	Ризик	Серйозність	Імовірність	Вплив	Профілактика	Наслідки
1	Лікарняний	1	0.7	0.7	Попередити про можливу відсутність	Працювати віддалено або попросити колег про допомогу
2	Проблеми з апаратурою	1	0.5	0.5	Перевіряти апаратуру перед початком робіт	Відкласти старт робіт до заміни апаратури
3	Дефекти, що блокують подальшу перевірку	5	0.3	1.5	Звертати увагу на проблемні зони та додавати додатковий час при плануванні на очікування вирішення можливих блокаторів	Затримка в завершенні тестування

План з документації наведено у таблиці 6.4

Таблиця 6.4 – Опис документації

Документація	Відповідальний	Дати
Набор тестів	Кочанов Максим	21.10 – 22.10
Тест план	Олександр Федюшин	23.10
Звіти з виявлених дефектив	Кочанов Максим	24.10 – 2.11
Звіт з результатами тестування	Олександр Федюшин	3.11

У ході тестування будуть використані наступні метрики :

- ефективність набору тестів : $\frac{\text{Кількість виявлених дефектів}}{\text{Кількість тестів у наборі}}$;

- покриття вимог тестами : $\frac{\text{Кількість тестів}}{\text{Кількість вимог}}$;

- рівень пропущених до релізу дефектів: $\frac{\text{Кількість виявлених дефектів після релізу}}{\text{Загальна кількість виявлених дефектів}}$;

- оцінка затрат часу : $\frac{\text{Передбачений час робіт}}{\text{Дійсний час робіт}}$;

- частка відхилених дефектів: $\frac{\text{Кількість дефектів не прийнятих для виправлення}}{\text{Загальна кількість дефектів}}$

7 ВИНЯТКИ З ЗАГАЛЬНОЇ МЕТОДОЛОГІЇ

У багатьох додатках використовуються унікальні технології , передбачити загрози для яких загальними методами майже не можливо . В даному разі слід спиратися на спеціалізовану літературу по даній сфері , а також на досвід спеціалістів з кібербезпеки.

У якості прикладу однієї з таких технологій розглянемо уразливості медичного протоколу DICOM.

DICOM (Digital Imaging and Communication in Medicine) [17] – протокол відображення медичних обстежень і передачі їх між різними компонентами системи. Цими компонентами можуть бути:

- медичне обладнання, яке безпосередньо робить сканування;
- DICOM-сервер - база даних для DICOM-файлів;
- DICOM-клієнт - зазвичай це додаток для перегляду результатів медичних обстежень.

Протокол DICOM має дві частини: опис формату файлу та опис мережевої взаємодії.

DICOM-файл - зображення медичного характеру, збережене в форматі DICOM. Цей формат - галузевий стандарт для зберігання і поширення медичних знімків.

Крім графічних даних, DICOM-файли можуть містити персональну інформацію у вигляді атрибутів, що дозволяють зіставити зображення з конкретною людиною і ідентифікувати пацієнта. До них відносяться стать, ім'я пацієнта, дата народження та ін. Список можливих атрибутів і їх опис можливо знайти в документації.

Мережева взаємодія у DICOM-протоколі описується за допомогою наступних команд :

Таблиця 7.1 – Основні мережеві команди DICOM-протоколу

Дія	Опис
C-ECHO	Тест з'єднання між двома пристроями
C-FIND	Пошук досліджень на віддаленому сервері
C-GET, C-MOVE	Завантаження досліджень з віддаленого сервера
C-STORE	Збереження дослідження на віддаленому сервері

Більшість хостів в мережі налаштовані таким чином, що будь-хто може встановити з ними з'єднання. Наприклад, можна розглянути процес отримання дослідження з віддаленого сервера з мережі. Для цього слід скористатись утилітами findscu і getscu з DCMTK - набору бібліотек і додатків, що реалізують більшу частину стандарту DICOM.

Першою командою отримуємо список всіх доступних досліджень на сервері.

$ Findscu -aet <AE Title> -P -k PatientName = "*" <host> <port>

З ключем -aet в запиті передаємо назву Application Entity Title. Зазвичай цей тайтл потрібен, коли слід розмежувати доступ до зображень при використанні одного DICOM-сервера різними PACS (Picture Archiving and Communication System - системи, в яких результати обстежень зберігаються в електронному вигляді). Це, свого роду, ідентифікація клієнта. Але проблема в тому, що тайтли на багатьох серверах налаштовані за замовчуванням. Це означає, підібрати його можливо, перебираючи значення за замовченням від різних розробників PACS і DICOM-серверів.

З ключем -k передаємо фільтр "PatientName = *", який дозволить показати будь-яке доступне дослідження на сервері.

Наступною командою викачуємо потрібне нам дослідження або все дослідження відразу.

$ Getscu -aet <AE Title> -P -k PatientName = "John Doe" <host> <port>

Значення ключів в команді аналогічні попередній.

Так за допомогою двох команд можна скачати дані з віддаленого DICOM-сервера при наявності однієї з умов: Application Entity Title не заданий або

Application Entity Title встановлений за замовчуванням, тому його вдалося підібрати.

Далі розглянемо популярні інструменти та реалізації протоколу і знайдені в них недоліки.

SimpleITK - реалізація протоколу, яка використовується в одному з великих проектів в області medical imaging NVIDIA CLARA.

Для пошуку вразливостей в ній використовувався фаззінг AFL зі словником. В результаті були виявлені переповнення купи. Спроби експлуатації призвели до виявлення більш простої уразливості - переповнення буфера через поле PatientName. Причому для переповнення буфера було досить створити файл з довжиною імені пацієнта понад 512 байт.

На даний момент вразливості вже виправлені.

DCMTK (DICOM Toolkit) - найстаріша реалізація DICOM-протоколу. Вона включає набір інструментів для роботи з протоколом: парсери DICOM-файлів в різних форматах і з різних форматів, а також утиліти для взаємодії з DICOM-сервером по мережі.

XXE в xml2dcm

Парсер xml2dcm конвертує результати медичного обстеження з формату XML в DICOM. У xml2dcm була знайдена вразливість до найпростішої XXE-атаки: створювався XML-файл з зовнішньої сутністю в імені пацієнта і на виході було отримано DICOM-файл з вмістом / etc / passwd.

На даний момент вразливість вже виправлена.

За аналогією з external entities в XML, сама по собі утиліта xml2dcm дозволяє створювати DICOM-файли з вмістом інших файлів всередині. Це зручно, тому що не потрібно писати дані в XML-файл - досить вказати в тезі PixelData шлях до файлу, з якого потрібно довантажити дані при конвертації.

Якщо в тезі PixelData вказати шлях до будь-якого файлу в системі, то після обробки утилітою xml2dcm ми отримаємо DICOM-файл з його вмістом.

Дана вразливість є актуальною, бо описану функцію неможливо відключити ніяким прапором , тому виправлення вимагає від розробника великі втрати ресурсів на зміну архітектури додатку.

DCMTK також надає реалізацію DICOM-сервера dcmqrscp. Тестування безпеки DICOM-сервера dcmqrscp було проведено за допомогою фаззінга AFLNet, в результаті якого було виявлено DoS.

На даний момент вразливість вже виправлена.

Також слід розглянути додаток ORTHANC. Цей продукт дуже простий в налаштуванні і використанні: він надає веб-обгортку для перегляду DICOM-файлів, і для роботи з DICOM-протоколом потрібно тільки браузер.

ORTHANC використовують в охороні здоров'я, в різних університетах і госпіталях, з його допомогою проводяться дослідження в області машинного навчання medical imaging.

Сервер ORTHANC відкритий до товариства розробників і надає REST API для написання різних плагінів. Якщо поглянути на список доступних методів, можна помітити методи для перезавантаження і виключення сервера, а також метод з назвою execute-script. Він приймає на вхід lua скрипти і без будь-якої валідації виконує їх на сервері.

Дана вразливість була виправлена в новій версії серверу, але у мережі ще можливо знайти Orthanc-сервери з працюючим методом execute-script.

Метод execute-script є небезпечним. Розробник ORTHANC вирішив цю проблему за допомогою аутентифікації, щоб виконувати такі запити могли тільки зареєстровані користувачі. Але за замовчуванням аутентифікація відключена при роботі з ORTHANC-сервером з офіційного сайту.

У докері від розробника вже включена аутентифікація за замовчуванням, але дана система аутентифікації вразлива до CSRF-атаці. Досить створити сторінку з наступним змістом і відправити її користувачеві:

```
<html>
  <body>
```

```
<form action="http://<host>:8042/tools/execute-script" method="POST" enctype="text/plain">
  <input type="hidden" name="cmd" value="'mkdir /tmp/testCSRF';os.execute(cmd)"/>
  <input type="submit" value="Submit request" />
 </form>
</body>
</html>
```

Коли користувач відкриває сторінку, на сервер відправляється команда, яка запускає виконання довільного коду.

Дана вразливість не була усунена , розробником була лише змінена технічна документація, що тепер попереджує майбутніх користувачів о наявній проблемі.

Як бачимо, знайти уразливості в medical imaging досить просто , що показує слабкий рівень захищеності таких технологій. Саме тому при розробці спеціалізованих додатків слід також не забувати про забезпечення безпеки, навіть якщо це вимагатиме додаткових ресурсів чи знань .

ВИСНОВОК

Останнім часом список сервісів і ресурсів ,які виросли в інтернеті, дуже поширився. Інтернет перетворився з одноманітних статичних сторінок в потужний інструмент інтерактивності і спілкування з кінцевими користувачами. Веб-додатки в даний час набули небувалу популярність тому що вони пропонують масу важливих переваг, які відсутні в звичайних додатках.

Кіберзлочинність зараз розвинена як ніколи – адже майже кожна компанія має свій сайт в інтернеті, а зловмисник у мережі може легко залишатися абсолютно анонімним. Для забезпечення основних принципів конфіденційності, цілісності, автентифікації, авторизації, доступності та неспростовності, тобто для створення цілісної безпечної системи організації часто витрачають багато коштів на тестування безпеки. Цей процес часто займає багато часу, та може не виправдати вкладених коштів. Наприклад , для маленьких компаній працюючих на закордонні закази часто не має можливості наймати професіоналів та проводити повний тест продукту. У рамках бюджету для компаній краще виявити основні загрози для їх продукту і протидіяти вже ним.

Так як існує велика кількість різноманітних загроз для веб додатків та методів для захисту від даних загроз, у даній роботі проводиться дослідження з метою виявлення найпоширеніших загроз, їх аналізу та розробці методів захисту саме від них, адже реалізувати повний комплекс захисту від усіх можливих атак на веб додаток неможливо через обмеження у фінансах та часі.

Найбільш поширеними загрозами для безпеки мобільних додатків за даними статистики є : ін'єкція, недоліки аутентифікації, розголошення конфіденційних даних, зовнішні сутності XML (XXE), недоліки контролю доступу, некоректна настройка параметрів безпеки, міжсайтове виконання сценаріїв (XSS), небезпечна десеріалізація, використання компонентів з відомими вразливостями, недоліки журналювання і моніторингу.

Для аналізу даних загроз було спершу обрано методику оцінки ризиків OWASP. Дана методика дозволяю отримати узагальнені данні про популярні загрози, але не інформацію про конкретні вразливості в реальних додатках. Основним мінусом даної методики слід вважати досить сильний фокус на факторі поширеності загрози.

Для вирішення цього недоліку була розроблена методика оцінки сукупного ризику. Методика оцінки сукупного ризику - це удосконалена методика оцінки ризику OWASP , яка на відміну від існуючої методики дозволяє оцінити загрози для мобільних додатків не за їх поширеністю, а за їх сукупним ризиком для додатку. Запропонований у даній методиці підхід також не враховує джерела загроз та технічні особливості окремих додатків, але формує пріоритет розробки захисту від загрози спираючись не на її поширеність, а на її сукупному ризику для додатку.

На основі виявленого переліку найпопулярніших загроз для веб додатків і застосування удосконаленої методології сукупного ризику був проведений детальний аналіз даних загроз, а також були виявлені фактори ризику, характерні для кожної із загроз. Ці фактори визначалися на основі доступної статистики.

Для повноцінного забезпечення безпеки веб додатку були досліджені нормативні документи з даної теми . Рекомендації по захисту веб додатків описані в НД ТЗІ 2.5-010-03. Даний документ описує різні норми щодо захисту веб додатку від загроз та функціональні послуги безпеки інформації.

Внаслідок детального аналізу переліку найнебезпечніших для веб додатків загроз , шляхів їх реалізації та державних стандартів з забезпечення безпеки, був розроблений комплекс рекомендацій для всієї команди проекту по захисту веб додатку від погроз.

Однією з ключових ролей при забезпеченні безпеки додатку є тестувальники. Тому був розроблений шаблон тест плану з проведення тестування безпеки веб додатку.

Розроблений тест-план має ряд переваг для проекту в цілому :

- добре розпланований, систематизований підхід дозволяє досягти кращих результатів, а також дозволяє виявляти більшу кількість помилок, ніж неорганізована, погано розпланована діяльність;
- тест план дозволяє управляти процесом тестування більш ефективно;
- тест план дозволяє побачити і зрозуміти мінімальний рівень тестування і отримати уявлення про рівень проведеного тестування кожної області продукту;
- тест план дозволяє досягти угоди між виконавцями, замовником і менеджером, про те, яким чином і в які терміни буде проводитися тестування.

Проте, слід мати на увазі, що в багатьох додатках використовуються унікальні технології , передбачити загрози для яких загальними методами майже не можливо . В даному разі слід спиратися на спеціалізовану літературу по даній сфері , а також на досвід спеціалістів з кібербезпеки.

При успішній та повної реалізації розроблених рекомендацій при розробці додатку, ризик для проектної команди розробити уразливий до проаналізованих загроз додаток та спричинити збитки замовнику через не усунені загрози безпеці додатку перестає існувати , так як розроблений додаток буде не вразливим до даних загроз безпеці на цей час.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аналітика [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2018/#id9.

2. Перелік загроз [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

3. Модель загрози [Електронний ресурс] – Режим доступу до ресурсу: https://www.owasp.org/index.php/Threat_Risk_Modeling.

4. Методологія [Електронний ресурс] – Режим доступу до ресурсу: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

5. SQL injection [Електронний ресурс] – Режим доступу до ресурсу: https://portswigger.net/web-security/sql-injection.

6. Shacklett M. Authentication [Електронний ресурс] / Mary Shacklett. – 2021. – Режим доступу до ресурсу: https://searchsecurity.techtarget.com/definition/authentication.

7. Відомості Верховної Ради України (ВВР). ЗАКОН УКРАЇНИ Про інформацію / Відомості Верховної Ради України (ВВР). – Київ, 1992. – 650 с.

8. Відомості Верховної Ради України (ВВР). ЦИВІЛЬНИЙ КОДЕКС УКРАЇНИ / Відомості Верховної Ради України (ВВР). – Київ, 2003. – 356 с.

9. XML external entity (XXE) injection [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://portswigger.net/web-security/xxe.

10. Access control vulnerabilities and privilege escalation [Електронний ресурс] // 2017 – Режим доступу до ресурсу: https://portswigger.net/web-security/access-control.

11. Security Misconfiguration [Електронний ресурс] – Режим доступу до ресурсу:https://www.manageengine.com/vulnerability-management/misconfiguration/.

12. Cross-site scripting [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://portswigger.net/web-security/cross-site-scripting.

13. Insecure deserialization [Електронний ресурс] // 2020 – Режим доступу до ресурсу: https://portswigger.net/web-security/deserialization.

14. Sass R. You Can’t Ignore Using Components With Known Vulnerabilities [Електронний ресурс] / Rami Sass. – 2018. – Режим доступу до ресурсу: https://www.whitesourcesoftware.com/resources/blog/using-components-with-known-vulnerabilities/.

15. Logging & Monitoring: definitions and best practices [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.vaadata.com/blog/logging-monitoring-definitions-and-best-practices/.

16. Департамент спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України. Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу / Департамент спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України. – Київ, 2003.

17. Недяк М. До чого призводять уразливості протоколу DICOM [Електронний ресурс] / Марія Недяк. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/company/bizone/blog/547586/