ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Зозуля Володимир Андрійович

УДК 004.056.5

Реалізація алгоритмічних та програмних методів захисту даних у комп'ютерних мережах

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавра

Нормоконтроль	Студент
_____Стисло О.В. (підпис, дата, розшифрування підпису)	_____Зозуля В.А. (підпис, дата, розшифрування підпису)
Допускається до захисту Завідувач кафедри	Керівник роботи
_____к.т.н., доц. Ващишак С.П. (підпис, дата, розшифрування підпису)	к.ф-м.н., доц. _____Остафійчук П. Г. (підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«___» _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Зозуля Володимир Андрійович

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Реалізація алгоритмічних та програмних методів захисту даних у комп'ютерни мережах

керівник роботи:

Остафійчук Петро Георгійович, к.ф-м.н., доц.

затверджена наказом вищого навчального закладу від «12» березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: Мова програмування Java

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Аналіз відомих програмних методів захисту даних в комп'ютерних мережах

2. Моделювання програмних методів захисту даних в комп'ютерних мережах

3. Реалізація структури алгоритму та методу шифрування

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд наявних програмних аналогів	15.01.2024	Виконано
2.	Розробка та моделювання програмних методів захисту даних в комп'ютерних мережах	29.01.2024	Виконано
3.	Розробка дизайну веб-сайту	10.02.2024	Виконано
4.	Реалізація структури та розробка програми для захисту даних	08.03.2024	Виконано
5.	Оформлення пояснювальної записки	11.04.2024	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	27.04.2024	Виконано

Студент _____ Зозуля В.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Остафійчук П.Г.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
13	Модулі інформаційної безпеки	26	Структура User Story
18	Система firewall	31	Методи проєктування ПЗ
20	Система Intrusion Detection Systems (IDS)	32	Патерни програмування
23	Система Deep Packet Inspection (DPI)	35	Архітектурний шаблон "Мікросервіси"

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці та реалізації ефективних алгоритмічних та програмних методів захисту даних у комп'ютерних мережах. Робота включає аналіз сучасних загроз безпеці в мережевому середовищі, визначення вразливості та потенційних атак, що можуть бути використані для несанкціонованого доступу до даних. На основі цього аналізу пропонується набір заходів та технологій для забезпечення конфіденційності, цілісності та доступності даних у мережевому середовищі.

У роботі також розглядаються сучасні методи шифрування, механізми аутентифікації та авторизації, системи виявлення та запобігання вторгнень, а також інші техніки та стратегії захисту даних. Зокрема, досліджується ефективність таких методів у реальних мережевих середовищах та їхній вплив на продуктивність та завантаження мережевих ресурсів.

Завдяки цій роботі отримуємо глибше розуміння проблем безпеки даних у мережевих системах та розробляємо практичні рекомендації та рішення для підвищення рівня безпеки та захисту приватності в інформаційних системах.

КЛЮЧОВІ СЛОВА: JAVA, КЛІЄНТ, ШИФРУВАННЯ.

SUMMARY

The qualification work is dedicated to the development and implementation of effective algorithmic and software methods for data protection in computer networks. The work includes an analysis of contemporary security threats in network environments, identification of vulnerabilities and potential attacks that could be used for unauthorized access to data. Based on this analysis, a set of measures and technologies is proposed to ensure the confidentiality, integrity, and availability of data in network environments.

The work also discusses modern encryption methods, authentication and authorization mechanisms, intrusion detection and prevention systems, as well as other techniques and strategies for data protection. Specifically, the effectiveness of such methods in real network environments and their impact on performance and resource utilization are investigated.

This work contributes to a deeper understanding of data security issues in network systems and develops practical recommendations and solutions to enhance the level of security and privacy protection in information systems.

KEY WORDS: website, JAVA,SHIFROWANNIA , client.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Html – мова гіпертекстової розмітки;
UI – інтерфейс користувача;
Css – каскадні таблиці стилів;
Scss – препроцесор мови Css;
Js – JavaScript.

ВСТУП

Актуальність теми. Захист даних в комп'ютерних мережах є надзвичайно актуальним в сучасному цифровому світі, оскільки зростаюча кількість даних, що обробляються та передаються через комп'ютерні мережі, створює надзвичайно великі ризики щодо їх безпеки.

З одного боку, зростаюча кількість інформації в мережах створює привабливу мету для хакерів, кіберзлочинців та зловмисників, які шукають способи несанкціонованого доступу до конфіденційних даних. Відомі приклади масштабних кібератак, таких як витік даних з великих корпорацій, підкреслюють необхідність ефективного захисту інформації у мережевих середовищах.

З іншого боку, інноваційні технології, такі як Інтернет речей (IoT), хмарні обчислення та машинне навчання, розширюють мережеве середовище і збільшують поверхню атак для потенційних загроз. Це робить проблему захисту даних у мережах ще більш актуальною та складною.

Отже, розробка та реалізація ефективних алгоритмічних та програмних методів захисту даних у комп'ютерних мережах стає надзвичайно важливим завданням для забезпечення конфіденційності, цілісності та доступності даних у цифровому віці. Робота в цьому напрямку має великий практичний і науковий потенціал для подальшого розвитку безпеки мереж та захисту інформації.

Мета і завдання дослідження. Мета дослідження полягає у розробці та реалізації ефективних алгоритмічних та програмних методів захисту даних у комп'ютерних мережах. Це дослідження ставить за мету забезпечення конфіденційності, цілісності та доступності інформації у мережевому середовищі.

Перше завдання дослідження полягає у проведенні аналізу сучасних загроз безпеці в комп'ютерних мережах. Визначаються потенційні вразливості та методи атак, які можуть бути використані для несанкціонованого доступу до

даних. Наступне завдання включає розгляд існуючих алгоритмічних та програмних методів захисту даних. Це охоплює методи шифрування, механізми аутентифікації та авторизації, системи виявлення та запобігання вторгнень, а також інші техніки захисту. Далі розробляються нові алгоритмічні підходи та програмні рішення для підвищення рівня безпеки даних у мережевому середовищі з урахуванням виявлених загроз та вразливостей. Після розробки методів захисту даних проводиться їхня реалізація у вигляді програмного забезпечення або інтеграції з існуючими системами управління мережами. Наступним кроком є експериментальне тестування та оцінка ефективності розроблених методів захисту даних у реальних мережевих середовищах, включаючи їх вплив на продуктивність та завантаження мережевих ресурсів. У завершенні проводяться висновки щодо результатів дослідження та розробляються практичні рекомендації для підвищення рівня безпеки та захисту даних у комп'ютерних мережах.

Об’єкт дослідження. Процес захисту даних у комп'ютерних мережах.

Предмет дослідження. Алгоритмічні та програмні методи захисту даних, їх реалізація та ефективність у контексті забезпечення безпеки інформації в комп'ютерних мережах.

Методи дослідження. Для вирішення поставленого завдання буде проведення літературного огляду, аналізу загроз та вразливостей, розробку нових методів захисту, експериментальне тестування, аналіз впливу на продуктивність та порівняльний аналіз.

Практичне значення одержаних результатів. Розроблені методи підвищують рівень безпеки даних, забезпечуючи надійний захист від несанкціонованого доступу та зловмисних атак. Це підвищує загальний рівень безпеки комп'ютерних мереж.

Структура роботи. Розділи – 3. Загальний обсяг основної частини – 61 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. АНАЛІЗ ВІДОМИХ ПРОГРАМНИХ МЕТОДІВ ЗАХИСТУ ДАНИХ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

1.1 Аналіз сучасних загроз безпеці програмного забезпечення в комп'ютерних мережах

Сучасний світ цифрових технологій приніс безліч можливостей і викликів, з якими повинні стикатися користувачі комп'ютерних мереж. З кожним днем відбувається значний розвиток та поширення інформаційних технологій, але разом з цим зростає й рівень загроз для безпеки цих мереж. Аналіз сучасних загроз безпеки в комп'ютерних мережах є критично важливим завданням для забезпечення захисту конфіденційності, цілісності та доступності даних індивідів, компаній та урядів.

У цьому розділі ми розглянемо різноманітні аспекти кібербезпеки, що становлять потенційні загрози для комп'ютерних мереж. Звернемо увагу на різноманітні види кібератак, включаючи віруси, фішинг, DDoS-атаки та інші, а також проаналізуємо можливі наслідки цих загроз. Крім того, ми дослідимо важливість застосування ефективних стратегій захисту, включаючи аутентифікацію, моніторинг мережі та навчання персоналу з питань кібербезпеки.

Проаналізувавши ці аспекти, ми матимемо можливість зрозуміти сучасні загрози безпеки в комп'ютерних мережах більш детально та розробити стратегії для їхнього виявлення, запобігання та управління.

Інформаційна безпека стає ключовим питанням для організацій усіх розмірів та сфер діяльності. У цьому контексті важливо розглянути різноманітні модулі, які складають інформаційну безпеку комп'ютерних систем.

Кожен модуль відіграє важливу роль у створенні комплексної системи захисту. Від аутентифікації та авторизації до захисту від вразливостей програмного забезпечення та фізичної безпеки - кожен з них є критичним

елементом, спрямованим на запобігання несанкціонованому доступу, збереження конфіденційності та забезпечення безпеки мереж усіх рівнів складності [1].

У цьому огляді ми докладемо зусиль, щоб розглянути кожен з цих модулів детальніше та з'ясувати, як вони взаємодіють між собою для забезпечення ефективної захищеності інформації. Розглядатимемо їхню важливість, принципи дії та способи впровадження для забезпечення безпеки даних та мереж у сучасному цифровому середовищі.

Класифікують сім основних модулів інформаційної безпеки (рис. 1.1):

1. Аутентифікація – важливим аспектом безпеки є впевненість, що лише правомірні користувачі мають доступ до системи. Аутентифікація використовується для перевірки ідентичності користувачів, зазвичай через паролі, біометричні дані або інші методи. Забезпечення надійного процесу аутентифікації є важливим для запобігання несанкціонованому доступу до системи.

2. Авторизація – після успішної аутентифікації система повинна визначити, які дії користувача допускаються. Модуль авторизації визначає права доступу користувачів до конкретних ресурсів або функціональностей системи. Це дозволяє контролювати, які дії можуть виконувати користувачі в межах системи.

3. Шифрування – шифрування використовується для захисту конфіденційності даних шляхом перетворення їх у незрозумілу форму. Тільки особи з правильним ключем можуть розшифрувати дані. Цей модуль гарантує, що навіть якщо зловмисники отримають доступ до зашифрованих даних, вони не зможуть прочитати їх.

4. Моніторинг та аудит безпеки – цей модуль відстежує дії користувачів та події в системі для виявлення аномальних або підозрілих активностей. Це допомагає виявити потенційні загрози та невдачі в безпеці, а також надає дані для подальшого аналізу та вдосконалення системи безпеки.

5. Захист від вразливостей програмного забезпечення - цей модуль орієнтований на виявлення та усунення вразливостей у програмах та операційних системах, що можуть бути використані зловмисниками для атак.

Регулярні оновлення та допомагають зменшити ризик використання таких вразливостей.

6. Фізична безпека – на більш вищому рівні, цей модуль охоплює заходи для захисту фізичного доступу до комп'ютерних систем та обладнання. Це може включати контроль доступу до серверних кімнат, використання біометричних сканерів або фізичний захист пристроїв від крадіжок.

7. Навчання та свідомість користувачів – один з найбільш вразливих елементів будь-якої системи безпеки – людський фактор. Цей модуль орієнтований на навчання користувачів щодо безпечних практик використання комп'ютерних систем та інформаційних ресурсів. Такі програми навчання допомагають усвідомити користувачам загрози та викласти правила безпеки.

Рисунок 1.1 – Модулі інформаційної безпеки

Модулі інформаційної безпеки взаємодіють між собою, утворюючи комплексну систему захисту, яка забезпечує повну безпеку інформації та мережі. Ось кілька способів, якими вони можуть взаємодіяти:

Спільна робота для забезпечення доступу: модулі аутентифікації та авторизації часто працюють разом. Після успішної аутентифікації модуль

авторизації визначає, які дії користувач може виконувати, на основі його ролі та прав доступу [2].

Використання шифрування для захисту даних: шифрування використовується для захисту інформації від несанкціонованого доступу. Аутентифікація дозволяє встановити, хто має доступ до даних, а шифрування забезпечує їхню конфіденційність у випадку, якщо доступ став несанкціонованим.

Моніторинг та аудит безпеки для виявлення загроз: модулі моніторингу та аудиту безпеки використовуються для відстеження дій користувачів та подій у мережі. Вони можуть співпрацювати з модулями аутентифікації та авторизації для виявлення підозрілих дій та вживання заходів захисту.

Захист від вразливостей програмного забезпечення: модулі захисту від вразливостей програмного забезпечення працюють разом з модулем моніторингу для виявлення та усунення потенційних вразливостей. Після виявлення вразливості можуть вживатися заходи для її усунення та впливу на систему [3].

Фізична безпека та захист від внутрішньої загрози: модулі фізичної безпеки та захисту від внутрішньої загрози можуть співпрацювати для запобігання несанкціонованому доступу до фізичних ресурсів системи, таких як серверні кімнати або пристрої, і для виявлення незвичайних дій співробітників.

Загалом, взаємодія між цими модулями допомагає створити комплексну систему захисту, яка забезпечує ефективний захист та мережі від різних загроз.

1.2 Загальний опис програми захисту даних у комп'ютерних мережах

Програма захисту даних у комп'ютерних мережах - це комплексний набір стратегій, інструментів та процедур, спрямованих на забезпечення конфіденційності, цілісності та доступності інформації, яка пересувається через мережу комп'ютерів. Основна мета такої програми - запобігання несанкціонованого доступу до даних, виявлення та вирішення потенційних

загроз безпеки, а також відновлення функціональності мережі у випадку інцидентів.

Програма захисту даних може включати в себе різноманітні складові, такі як аутентифікація та авторизація користувачів, шифрування даних, моніторинг та аудит безпеки, захист від вразливостей програмного забезпечення, фізична безпека обладнання та заходи захисту від внутрішніх загроз [4].

Ця програма ретельно розробляється з урахуванням специфіки і потреб конкретної мережі та організації. Вона вимагає постійного оновлення та вдосконалення, оскільки загрози безпеки постійно еволюціонують. Ефективна програма захисту даних дозволяє забезпечити високий рівень безпеки інформації, збереження довіри користувачів та безперебійне функціонування комп'ютерних мереж у будь-яких умовах.

Різні мережі мають свою унікальну специфіку і відповідні потреби щодо захисту даних. Ось огляд найбільш поширених типів мереж і їхніх особливостей:

1. Корпоративні мережі - корпоративні мережі зазвичай використовуються в офісних середовищах для спільної роботи та обміну даними між співробітниками. Їхні головні потреби включають в себе високу доступність, швидкість передачі даних і захист від зовнішніх загроз, таких як кібератаки та віруси.

2. Хмарні мережі – хмарні мережі базуються на віртуальних ресурсах, що знаходяться в розподіленому середовищі. Вони вимагають високого рівня безпеки для захисту конфіденційної інформації, що зберігається в хмарі, а також забезпечення надійного доступу до цих даних для авторизованих користувачів.

3. Мобільні мережі - мобільні мережі орієнтовані на забезпечення зв'язку та доступу до даних для користувачів, які пересуваються. Вони вимагають високого рівня безпеки для захисту даних, які передаються через бездротові канали, а також захисту від загроз, що походять з мережі.

4. Інтернет-мережі – інтернет-мережі мають широкий спектр користувачів і вимагають високого рівня безпеки для захисту даних, що пересилаються через

відкриті канали зв'язку. Захист від кібератак, фішингу та вірусів є критично важливим для забезпечення безпеки користувачів у цьому середовищі.

Кожен з цих типів мереж має свої особливості та вимоги до захисту даних, і важливо розробляти програми захисту, які враховують ці унікальні потреби.

Загрози в комп'ютерних мережах є різноманітними і постійно еволюціонують з розвитком технологій. Класифікують кілька потенційних загроз, з якими можуть стикатися комп'ютерні мережі:

– кібератаки включають в себе різноманітні види атак, такі як DDoS (розподілений деніальний сервіс), фішинг, введення SQL-запитів та інші, які спрямовані на отримання несанкціонованого доступу до даних або завдання шкоди мережі;

– віруси, черв'яки, троянські програми ці зловмисні програми можуть використовуватися для пошкодження файлів, крадіжки конфіденційної інформації, а також для створення бот-мереж, які використовуються для виконання атак масового спаму або DDoS;

– вразливості програмного забезпечення: Незакриті вразливості в операційних системах, програмному забезпеченні та додатках можуть бути використані зловмисниками для внедрення шкідливого коду, крадіжки даних або отримання несанкціонованого доступу;

– неаутентичні користувачі – ці внутрішні загрози можуть включати неаутентичних або недбалих користувачів, які можуть випадково або навмисно викликати проблеми в мережі, втратити або пошкодити дані;

– соціально-інженерні атаки спрямовані на використання маніпуляції або обману людей з метою отримання конфіденційної інформації або надання доступу до системи;

– втрата даних - якщо дані не захищені належним чином, вони можуть стати предметом крадіжки або витоку, що може призвести до серйозних фінансових, репутаційних та юридичних наслідків для організації;

– фізичні загрози включають в себе крадіжку або пошкодження фізичного обладнання, такого як сервери або мережеві пристрої, а також можливість негативного впливу на мережу через природні катастрофи.

Забезпечення відповідного захисту мережі включає в себе використання різноманітних стратегій, інструментів та процедур, які мінімізують ризики і реагують на можливі загрози [5].

1.3 Розгляд та аналіз існуючих алгоритмічних та програмних методів захисту даних

У сучасному цифровому світі захист конфіденційності, цілісності та доступності даних стає все більшою пріоритетною задачею для організацій і приватних осіб. Зростання кількості та складності кіберзагроз, а також висока цінність інформації, робить важливим розуміння та використання ефективних алгоритмічних та програмних методів захисту даних.

У цьому розділі ми розглянемо різні методи захисту даних, що включають в себе як класичні алгоритмічні підходи, так і сучасні програмні рішення. Ми розглянемо їхні переваги та недоліки, а також обговоримо сфери їх застосування та потенційні виклики.

Цей аналіз дозволить нам краще зрозуміти, які методи захисту даних найбільш ефективні в різних ситуаціях, і як можна оптимізувати стратегії безпеки для досягнення максимального рівня захисту в цифровому середовищі.

Перед розробкою програмного забезпечення для захисту даних було проаналізовано такі аналоги:

– Firewall;
– Intrusion Detection Systems (IDS);
– Deep Packet Inspection (DPI).

Розглянувши дані веб-ресурси можна побачити сильні та слабкі сторони.

1. Система фаєрволу (firewall) є ключовим елементом в сфері інформаційної безпеки, який призначений для контролю трафіку мережі, що

входить та виходить з комп'ютерної мережі. Основною метою фаєрволу є захист комп'ютерних систем від несанкціонованого доступу та інших кіберзагроз, забезпечення цілісності даних і безпеки мережі (рис. 1.2).

Використання цієї системи рекомендується для будь-якої комп'ютерної мережі для забезпечення її захисту від різних кіберзагроз. Його основне завдання — контролювати та фільтрувати вхідний і вихідний мережевий трафік на основі заздалегідь визначених правил безпеки. Фаєрволи можуть бути як апаратними пристроями, так і програмними додатками. Фаєрволи є важливим компонентом комплексної стратегії кібербезпеки і відіграють ключову роль у захисті інформаційних систем від різноманітних загроз.

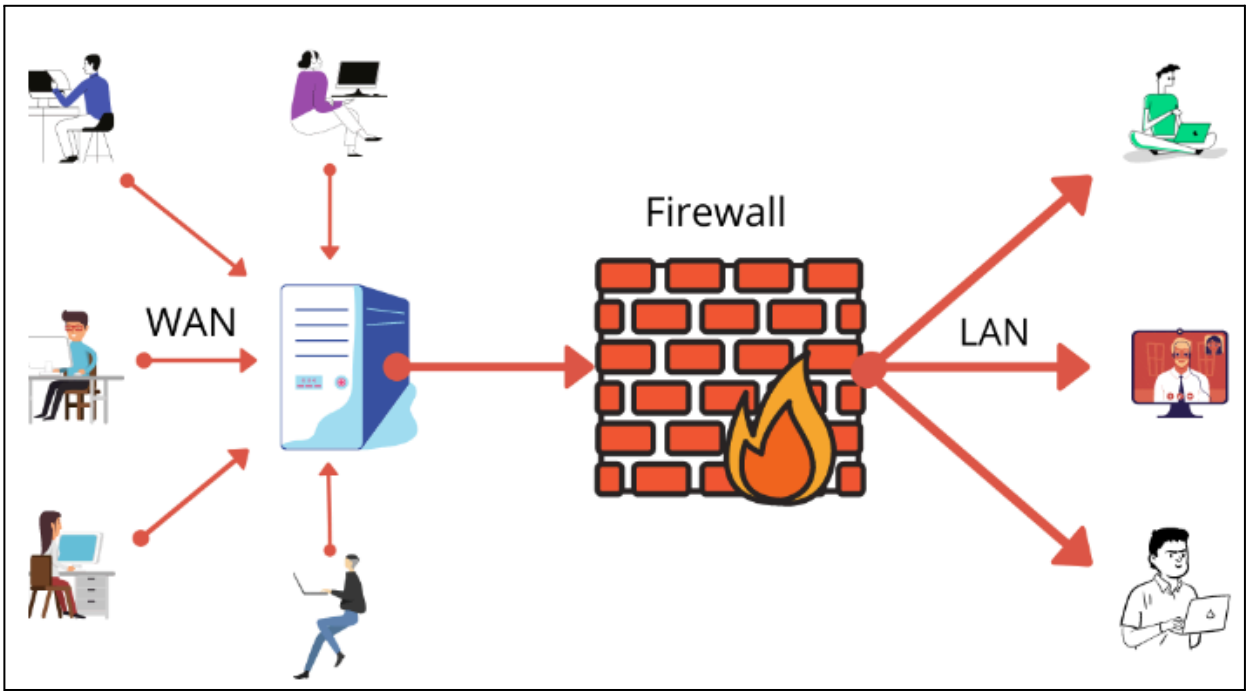


Рисунок 1.2 – Система firewall

Для кращого розуміння даної системи на рисунку 1.2 зображений принцип роботи firewall. Ось деякі ключові аспекти системи фаєрволу:

– фаєрвол контролює трафік, що входить та виходить з мережі, на основі заданих правил та критеріїв. Це дозволяє блокувати або дозволяти певний трафік відповідно до встановлених політик безпеки;

– фаєрвол може блокувати спроби несанкціонованого доступу до мережі або комп'ютерів, використовуючи різні методи, такі як фільтрація за IP-адресами, портами або протоколами;

– фаєрвол може виявляти та блокувати шкідливе програмне забезпечення, таке як віруси, троянські програми та шпигунське ПЗ, перед тим як воно зможе вторгнутися в мережу або систему;

– фаєрвол може вести журнали та аналізувати мережеву активність, щоб виявляти потенційні загрози та незвичайну поведінку;

– фаєрвол може відстежувати та керувати мережевим трафіком залежно від заданих правил, що дозволяє оптимізувати пропускну здатність та ефективно використовувати ресурси мережі;

– фаєрвол може бути як апаратним пристроєм, так і програмним забезпеченням, яке встановлюється на серверах або маршрутизаторах мережі.

Незважаючи на свою важливість і ефективність, системи фаєрволу мають деякі недоліки та слабкі місця:

– деякі типи атак можуть обійти захист, наданий фаєрволом. Наприклад, атаки через злам портів або використання шкідливих програм, що використовують незахищені канали, можуть пройти через фаєрвол;

– неправильно налаштований фаєрвол або використання неадекватного апаратного обладнання може призвести до сповільнення швидкості мережі через затримки при обробці пакетів даних;

– фаєрволи можуть виявляти нормальний мережевий трафік як потенційно загрозливий, що може призводити до втрати легітимних даних або послуг;

– налаштування та керування фаєрволом може бути складним завданням, особливо для користувачів з обмеженим досвідом у сфері мережевої безпеки;

– фаєрвол може стати одним точковим вразливим пунктом у випадку атаки, якщо атакувач може проникнути через нього, то він може мати доступ до всієї мережі;

– розгортання та підтримка фаєрволу може бути витратним процесом, особливо для великих організацій або підприємств;

– фаєрволи, як правило, зосереджені на захисті мережі від зовнішніх атак, тому вони можуть бути не так ефективними у виявленні або запобіганні атакам внутрішньої загрози.

2. Система виявлення вторгнень (Intrusion Detection System, IDS) є важливим інструментом в області кібербезпеки, спроектованим для виявлення небажаної, підозрілої або зловмисної активності в мережі чи на комп'ютерних системах (рис.1.3). Основна мета IDS полягає в сповіщенні адміністраторів про потенційні загрози або вторгнення, що дозволяє їм реагувати та вживати заходів для захисту мережі та даних. Ось деякі ключові аспекти системи IDS:

– виявлення підозрілої активності: IDS аналізує мережевий трафік, системні журнали та інші джерела інформації для виявлення аномальної або незвичайної активності, яка може свідчити про потенційне вторгнення або атаку;

– аналіз сигнатур і зразків: IDS використовує бази даних сигнатур або зразків для виявлення відомих атак або шаблонів поведінки, що відповідають відомим загрозам;

– реакція на виявлені загрози: IDS може надсилати сповіщення адміністраторам або автоматично запускати заходи для блокування або ізоляції джерела вторгнення з мережі;

– різні типи IDS: існують два основних типи IDS: сигнатурний (заснований на виявленні відомих шаблонів атак) та аномалійний (заснований на виявленні аномальної поведінки);

– мережевий та хост-базовий IDS: мережеві IDS аналізують мережевий трафік, тоді як хост-базові IDS аналізують дані та активність на конкретній комп'ютерній системі;

– локальні та розподілені системи IDS: локальні системи IDS встановлюються на окремих комп'ютерах або серверах, тоді як розподілені системи IDS можуть моніторити великі мережі або сегменти мережі [6].

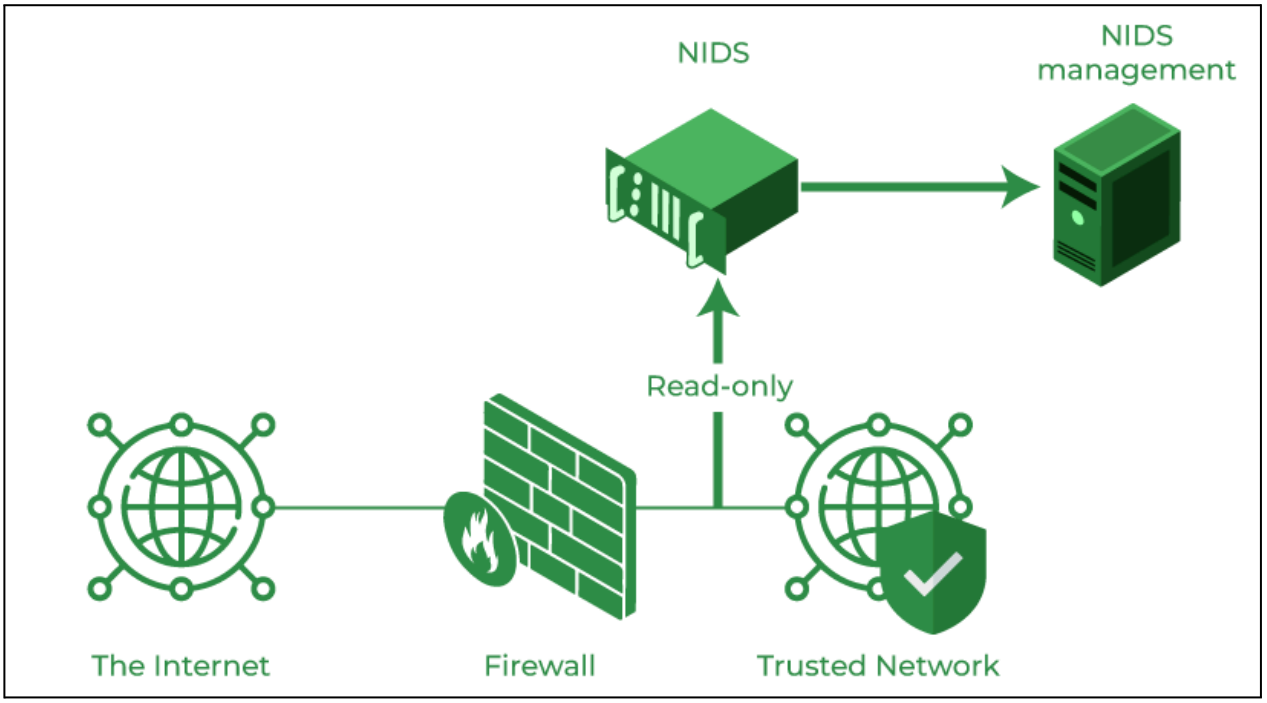


Рисунок 1.3 – Система Intrusion Detection Systems (IDS)

Незважаючи на свою корисність, системи виявлення вторгнень також мають деякі недоліки та слабкі місця:

– IDS можуть генерувати велику кількість ложнопозитивних сповіщень про нормальну мережеву або системну активність, що може перенавантажувати адміністраторів та призводити до ігнорування справжніх загроз;

– ложні негативи: Іноді IDS можуть пропустити атаки, які вони не здатні виявити через недостатню обчислювальну потужність або базу сигнатур;

– деякі атаки можуть обхід захист, зокрема ті, що використовують нові або невідомі вразливості, які не включені до бази сигнатур IDS;

– IDS вимагають налаштування, підтримки та постійного моніторингу для забезпечення ефективності. Недостатнє налаштування може призвести до недооцінки загроз, тоді як неправильне конфігурування може призвести до перевеликого обсягу ложнопозитивних сповіщень;

– IDS можуть виявляти лише те, що вони програмовані виявляти, а не завжди розуміють контекст атаки або поведінки мережі, що може призвести до пропускання підступних атак;

– постійне моніторування мережі може потребувати значних обчислювальних ресурсів, особливо для великих мереж або з'єднання;

– хоча деякі IDS можуть сповіщати про атаки майже в реальному часі, інші можуть мати затримки у виявленні та реакції на загрози;

– іноді сигнатури атак можуть застаріти або не оновлюватися вчасно, що може призвести до пропускання нових загроз.

Враховуючи ці недоліки, важливо використовувати системи виявлення вторгнень як частину комплексної стратегії кібербезпеки, доповнюючи їх іншими засобами захисту мережі.

3. Deep Packet Inspection (DPI) - це технологія аналізу пакетів даних, що передаються по мережі, на рівні їхнього вмісту та заголовків (рис. 1.4). Вона дозволяє детально аналізувати трафік і виявляти різноманітні аспекти комунікацій, включаючи типи протоколів, додатки, вміст і навіть потенційно шкідливий код. Ось деякі ключові аспекти системи Deep Packet Inspection:

– аналіз заголовків та вмісту пакетів: DPI здатна проникнути на глибину в мережевий трафік і аналізувати як заголовки, так і вміст кожного пакета даних;

– виявлення протоколів та додатків: DPI може розпізнавати різні типи мережевих протоколів і визначати, які додатки або сервіси використовуються для передачі даних;

– фільтрація трафіку: DPI дозволяє встановлювати правила фільтрації трафіку на основі різних параметрів, таких як адреса відправника, адреса отримувача, тип протоколу, вміст тощо;

– виявлення загроз та інцидентів безпеки: DPI може виявляти потенційні загрози безпеки, такі як атаки внутрішньої мережі, спроби вторгнення або витік конфіденційної інформації;

– блокування шкідливих вірусів і програм: DPI може блокувати відомі шкідливі програми, віруси та інші загрози безпеки перед тим, як вони зможуть нанести шкоду мережі або комп'ютерам;

– моніторинг і журналювання: DPI може вести журнали та моніторити мережеву активність для подальшого аналізу та виявлення потенційних проблем або вразливостей;

– використання у різних областях: DPI може бути застосована в різних галузях, включаючи мережеву безпеку, моніторинг мережевого трафіку, оптимізацію мережевих ресурсів та забезпечення дотримання правил корпоративної політики.

Хоча DPI має багато переваг, включаючи глибокий аналіз мережевого трафіку і виявлення складних загроз, вона також має деякі недоліки, такі як високі вимоги до обчислювальних ресурсів та можливість порушення приватності даних користувачів. Використання DPI повинно бути збалансованим і враховувати потреби та обмеження конкретної мережі чи системи [7].

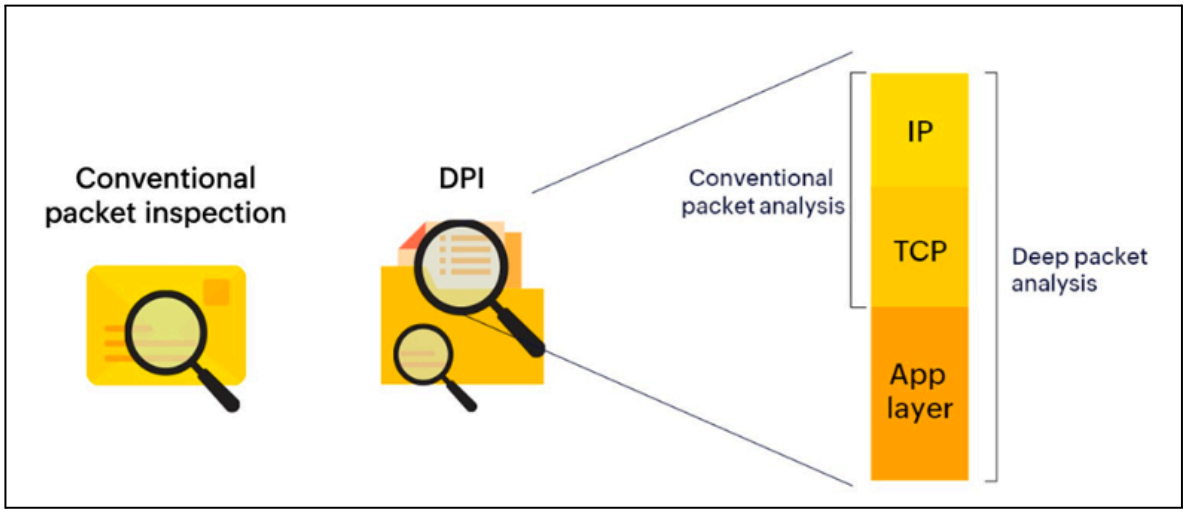


Рисунок 1.4 – Система Deep Packet Inspection (DPI)

Незважаючи на свою ефективність, система Deep Packet Inspection має деякі недоліки та слабкі місця:

– оскільки DPI може аналізувати вміст пакетів даних, існує ризик порушення приватності користувачів, особливо коли мова йде про персональні або конфіденційні дані;

– глибокий аналіз пакетів даних може потребувати значних обчислювальних ресурсів, особливо великих обсягів мережевого трафіку, що

може призвести до зниження продуктивності мережі або збільшення витрат на апаратне обладнання;

– деякі атаки можуть обхідати захист DPI, зокрема зашифрований трафік або використання альтернативних комунікаційних каналів, що робить систему вразливою до нових атак або розробок;

– провайдери мережі можуть блокувати або обмежувати доступ до певних видів DPI, що може перешкоджати використанню цієї технології для захисту мережі;

– налаштування і підтримка системи DPI може бути складною задачею, особливо для організацій з обмеженими ресурсами або обмеженим досвідом в цій спеціалізованій області;

– відсутність стандартизації у сфері DPI може призвести до суміші різних реалізацій, що може ускладнити взаємодію між різними системами та пристроями;

– деякі аспекти аналізу DPI можуть призводити до виникнення ложнопозитивних сигналів або помилкових сповіщень про загрози, що може перенавантажити адміністраторів і знизити довіру до системи.

Загалом, система Deep Packet Inspection є потужним інструментом для моніторингу та захисту мережі, але вона має свої обмеження та недоліки, які потрібно враховувати при її використанні [8].

1.4 Постановка задачі

Основною метою цієї дипломної роботи є розробка, реалізація та аналіз ефективних алгоритмічних та програмних методів захисту даних у комп'ютерних мережах. Очікується, що результати дослідження та реалізації методів захисту даних у комп'ютерних мережах допоможуть підвищити рівень безпеки мережевих інфраструктур. Особлива увага буде приділена аналізу ефективності та надійності розроблених методів в порівнянні з існуючими рішеннями. Для досягнення цих цілей необхідно вирішити наступні завдання:

– проаналізувати переваги та недоліки вже існуючих систем захисту даних в комп'ютерних мережах;

– вибрати інструменти розробки та технології програмування;

– розробити нові алгоритмічні методи захисту даних, які враховують сучасні кіберзагрози та виклики;

– реалізувати програмне забезпечення на основі розроблених алгоритмів для захисту даних у комп'ютерних мережах;

– протестувати та провести аналіз ефективності розроблених методів захисту на різних сценаріях та умовах.

Висновки до розділу 1

У цьому розділі було проведено аналіз наявних систем захисту даних в комп'ютерних мережах. Після пошуку відповідних програмних рішень та збору інформації були надані основні відомості про алгоритми та архітектури систем захисту даних в різних мережах, а також описані всі їхні переваги та недоліки які в подальшому будуть враховані при розробленні програмного забезпечення для захисту даних в комп'ютерних мережах. Для більшого розуміння архітектури та роботи наведених аналогів у розділі були додані зображення.

Під час подальшої розробки продукту, проаналізовано різні архітектури та побудову алгоритмів захисту даних, а також враховані всі недоліки які були проаналізовані в даному розділі. Це спрощує оптимізацію та проектування продукту.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ПРОГРАМНИХ МЕТОДІВ ЗАХИСТУ ДАНИХ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

2.1 Написання User Stories

User Stories є важливим інструментом у розробці програмного забезпечення, допомагаючи командам розробників розуміти, що саме потрібно враховувати при створенні продукту, і спрямовуючи їхні зусилля на відповідні завдання.

User Stories – це короткі описи функціональності програмного продукту, які формулюються з точки зору користувача. Кожна User Story описує конкретний сценарій використання програми або функціональну вимогу, яка відображає потреби користувача. Вони допомагають команді розробників та зацікавленим сторонам краще зрозуміти, що саме потрібно враховувати в процесі розробки програмного забезпечення (рис. 2.1).

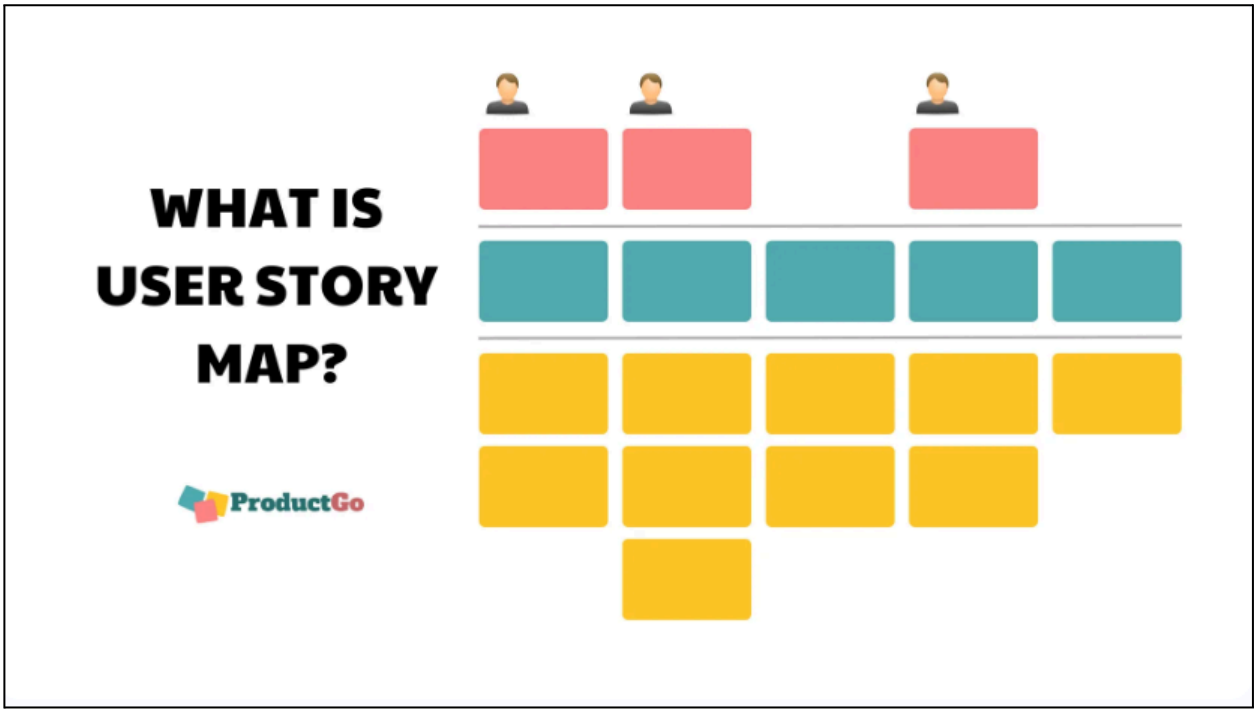


Рисунок 2.1 – Структура User Story

Кожна User Story має специфічний формат, що включає в себе:

1. Користувача: це особа або роль, яка буде взаємодіяти з продуктом.
2. Дію: конкретна дія, яку користувач планує виконати або функціональну вимогу, яку він хоче бачити реалізованою.
3. Мету: очікуваний результат або користь, яку користувач отримає в результаті виконання цієї дії або вимоги.

User Stories допомагають розробці програмного забезпечення таким чином:

1. Фокус на потребах користувача: вони допомагають зосередитися на тому, що саме важливо для користувача, допомагаючи уникнути зайвих функцій або затримок у випуску програми.
2. Зручна комунікація: вони створюють зрозумілу мову для всіх учасників проекту, забезпечуючи чітке спілкування між розробниками та клієнтами.
3. Пріоритизація роботи: вони допомагають команді визначити, які функції або вимоги мають вищий пріоритет, що сприяє ефективному розподілу ресурсів та розвитку продукту.
4. Валідація функціональності: вони дозволяють перевірити, чи вдачно реалізована певна функціональність програми шляхом перевірки, чи задовольняє вона очікування користувача.

Для програми шифрування даних в комп'ютерних мережах були визначені такі аспекти User Story:

1. Можливість налаштування та використання сильних алгоритмів шифрування: адміністратор мережі, розглядаючи безпеку передачі конфіденційної інформації через мережу, прагне мати можливість встановлювати та використовувати сучасні алгоритми шифрування, такі як AES (Advanced Encryption Standard), для забезпечення максимальної захищеності даних.
2. Двофакторна аутентифікація для безпеки доступу: користувачі, прагнучи підвищити рівень безпеки свого доступу до мережевих ресурсів, очікують можливості використання двофакторної аутентифікації, яка забезпечить захист навіть у випадку, якщо зловмисник дізнається пароль.

3. Засоби для виявлення та блокування несанкціонованого доступу: адміністратор, борючись з потенційними загрозами, хоче мати в своєму розпорядженні інструменти для миттєвого виявлення та блокування спроб несанкціонованого доступу до мережевих пристроїв, що можуть загрожувати безпеці даних.

4. Автоматичне резервне копіювання даних: користувачі бажають мати можливість автоматичного створення резервних копій своїх даних на захищених серверах з метою запобігання втратам у випадку виникнення непередбачених подій, таких як вірусні атаки або випадки збоїв у обладнанні.

5. Моніторинг та аналіз мережевого трафіку: адміністратор очікує на наявність засобів для постійного моніторингу та аналізу мережевого трафіку для виявлення підозрілої активності та потенційних загроз безпеці даних, що дозволить реагувати на них своєчасно.

6. Шифрування файлів перед передачею: користувачі очікують можливості захистити свої дані від перехоплення шляхом шифрування файлів перед їх передачею через мережу, що забезпечить конфіденційність інформації навіть у випадку несанкціонованого доступу.

7. Централізоване управління правами доступу: адміністратор, для забезпечення ефективного контролю над безпекою даних, очікує на можливість централізованого управління правами доступу до мережевих ресурсів, що дозволить точно налаштовувати рівні доступу для користувачів.

8. Автоматичне оновлення програмного забезпечення: користувачі хочуть, щоб їхні мережеві пристрої автоматично оновлювали програмне забезпечення з метою забезпечення захисту від відомих вразливостей та потенційних атак.

9. Аналіз журналів подій для виявлення аномальної активності: адміністратор очікує на наявність інструментів для аналізу журналів подій з метою виявлення та відстеження аномальної активності у мережі, що допоможе запобігти потенційним загрозам.

10. Надсилання зашифрованих повідомлень: користувачі хочуть мати

можливість надсилати зашифровані електронні листи та повідомлення через захищені канали зв'язку з метою збереження приватності та конфіденційності інформації.

User Stories є потужним інструментом у розробці програмного забезпечення, оскільки вони дозволяють командам розробників зосередитися на потребах користувачів і ефективно взаємодіяти з клієнтами та зацікавленими сторонами. Ці короткі, зрозумілі описи функціональності допомагають визначити, що саме має бути реалізовано в програмному продукті, і уникнути зайвих функцій або затримок у випуску. Вони сприяють зручній комунікації всередині команди розробників та з клієнтами, а також допомагають у визначенні пріоритетів та валідації функціональності програми. Загалом, використання User Stories сприяє покращенню якості програмного забезпечення та задоволенню потреб користувачів [9].

2.2 Проектування алгоритму та методів обробки даних в мережах

Проектування алгоритмів та методів обробки даних в мережах є ключовою складовою у сучасній інформаційній технології, оскільки дозволяє забезпечити ефективну передачу, обробку та захист інформації у мережевих середовищах. З розвитком комп'ютерних технологій та зростанням обсягу даних, що обробляються в мережах, стає все важливіше розробляти ефективні алгоритми та методи, що забезпечують оптимальну швидкість, безпеку та надійність обробки інформації.

У цьому контексті проектування алгоритмів та методів обробки даних в мережах включає в себе вивчення та розробку ефективних алгоритмів для різноманітних завдань, таких як маршрутизація даних, керування потоками, забезпечення безпеки та конфіденційності, виявлення та виправлення помилок, а також оптимізація роботи мережевих пристроїв [10].

У даному дослідженні ми розглянемо основні принципи та методи проектування алгоритмів та методів обробки даних в мережах, а також їхнє

застосування для вирішення актуальних завдань у сфері інформаційних технологій. Вивчення цієї теми допоможе зрозуміти важливість розробки ефективних алгоритмів для оптимізації роботи мереж та забезпечення безпеки та надійності обробки даних.

Ось детальніші пояснення про кожен з методів проєктування програмного забезпечення (рис. 2.2):

1. Метод Waterfall передбачає послідовне виконання фаз проєктування: вимог, дизайну, реалізації, тестування та впровадження. Кожна фаза починається лише після завершення попередньої. Цей метод відображається на водоспаді, де робота починається з верхнього потоку і спускається вниз. Він часто використовується для проєктів з визначеними та стабільними вимогами, які не потребують частих змін.

2. Ітеративний метод передбачає поділ розробки на короткі цикли, відомі як ітерації. Кожна ітерація включає в себе всі фази життєвого циклу проєктування, від визначення вимог до випуску. Кожна нова ітерація додає функціональність до продукту, що дозволяє швидко реагувати на зміни та внесення коректив.

3. Інкрементальний метод передбачає послідовне додавання функціональності до програми шляхом інкрементальних виправлень. Кожен інкремент може включати в себе нові функції або покращення функціональності, що вже існує. Цей метод дозволяє поступово розвивати програму, зберігаючи її працездатність на кожному етапі.

4. Спіральний метод поєднує в собі елементи каскадного та ітеративного проєктування. Проєкт рухається по спіралі, де кожен оберт включає в себе цикл ітерації, що включає в себе аналіз ризиків, планування, виконання та оцінку. Цей метод особливо корисний для проєктів з великою кількістю невизначеності та ризиків.

5. Agile є підхідом до розробки програмного забезпечення, який покладається на гнучкість, колаборацію та реагування на зміни. Основні принципи Agile включають ітеративність, інкрементальність, взаємодію з

клієнтом та самоорганізацію команди. Метод Agile дозволяє розробникам швидко реагувати на зміни в вимогах та максимально задовольняти потреби клієнта.

Кожен із цих методів має свої особливості та переваги, і вибір конкретного підходу залежить від потреб проєкту та вимог клієнта.

Для програми захисту даних в комп'ютерних мережах може бути найбільш підходящим метод ітеративного проєктування. Ось чому:

1. Короткі цикли розробки: цей метод дозволяє швидко створювати та випускати нові версії програмного забезпечення через короткі ітерації. Для захисту даних важливо швидко реагувати на нові загрози та вразливості.

2. Гнучкість та адаптивність: ітеративний підхід дозволяє змінювати пріоритети та функціонал під час розробки, що особливо важливо для програм захисту даних, де потрібно швидко реагувати на зміни в сфері кібербезпеки.

3. Залучення клієнта та фідбек: часті ітерації дозволяють активно залучати клієнта або користувачів до процесу розробки, що дозволяє забезпечити, що програмне забезпечення відповідає їхнім потребам і очікуванням.

4. Випробування та валідація: кожна ітерація може включати в себе тестування та валідацію нових функцій, що допомагає забезпечити високу якість та надійність програмного забезпечення для захисту даних.

Хоча інші методи також можуть бути використані для розробки програми захисту даних, метод ітеративного проєктування відповідає особливостям цього типу програмного забезпечення, забезпечуючи гнучкість, швидкість реакції та високу якість. Кожна ітерація (цикл) передбачає внесення змін на основі отриманих відгуків та нових даних, що дозволяє поступово покращувати кінцевий продукт. Цей метод широко застосовується в різних сферах, включаючи інженерію, програмну розробку, дизайн та бізнес-процеси. Метод ітеративного проєктування забезпечує постійне вдосконалення продукту, знижує ризики та підвищує задоволеність користувачів, що робить його ефективним підходом для багатьох видів проєктів. [9].

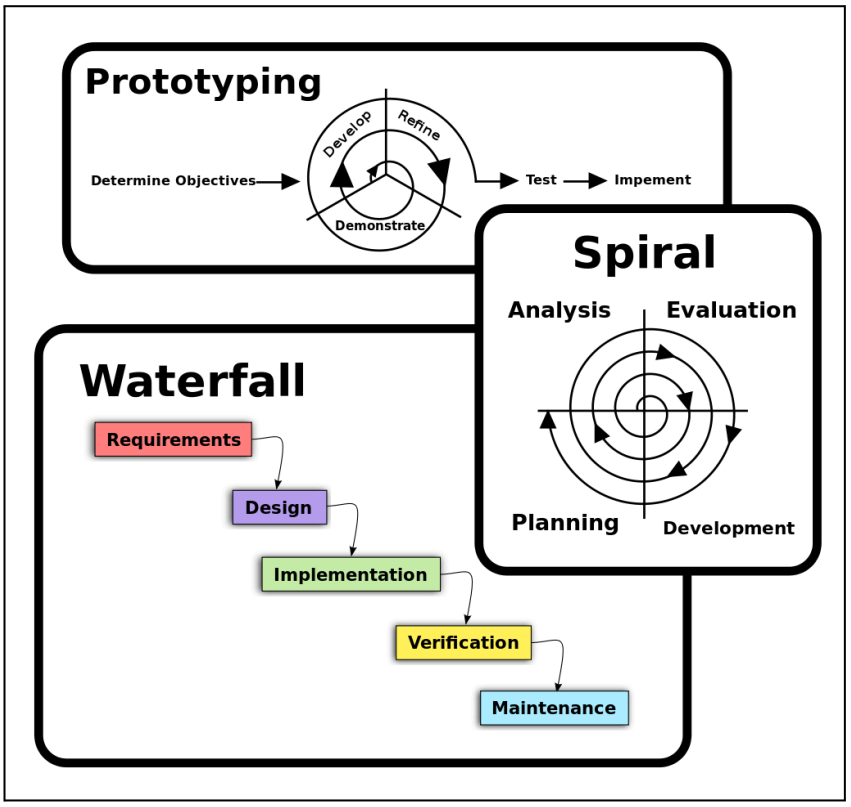


Рисунок 2.2 - Методи проєктування ПЗ

Патерни програмування (Design Patterns) - це загальні визначені рішення для типових проблем у розробці програмного забезпечення. Вони є відображенням кращих практик, які допомагають розробникам створювати ефективне та легко розширюване програмне забезпечення. Ось деякі типи паттернів програмування:

1. Породжувальні (Creational) патерни: ці патерни допомагають у створенні об'єктів та класів у системі. Деякі з найвідоміших породжувальних паттернів включають Singleton, Factory, Builder та Prototype.

2. Структурні (Structural) патерни: ці патерни визначають структуру класів та об'єктів у програмі. Вони допомагають покращити співпрацю між об'єктами та відображають спосіб їх групування. Деякі структурні патерни включають Adapter, Decorator, Facade та Bridge.

3. Поведінкові (Behavioral) патерни: ці патерни визначають взаємодію між об'єктами та способи організації алгоритмів. Вони дозволяють виокремити алгоритми з класів, щоб забезпечити більшу гнучкість та перевикористання коду.

До поведінкових патернів відносяться Observer, Strategy, Command та Iterator.

Патерни програмування дозволяють розробникам використовувати досвід та кращі практики, вже знайдені в галузі програмного забезпечення. Вони спрощують розробку, розуміння та підтримку коду, а також забезпечують стандартизацію та покращення якості програмного продукту (рис. 2.3).

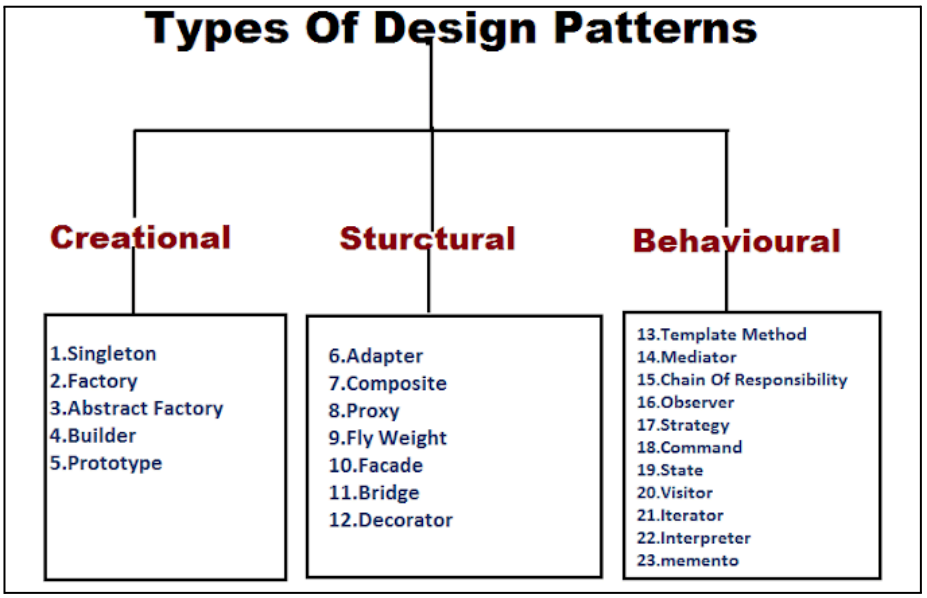


Рисунок 2.3 - Патерни програмування

Для розробки програми для захисту даних в комп'ютерних мережах може бути корисним використання патерна "Заступник" (Proxy).

Патерн "Заступник" дозволяє створити об'єкт-замінник або посередника, який контролює доступ до іншого об'єкта. Цей патерн особливо корисний для ситуацій, коли потрібно обмежити доступ до певних ресурсів або функціональності. Він може бути застосований для забезпечення безпеки в мережевих додатках, контролю доступу до конфіденційної інформації та фільтрації небезпечних запитів чи даних.

У контексті захисту даних в комп'ютерних мережах, патерн "Заступник" може використовуватися для наступних цілей:

1. Контроль доступу: об'єкт-заступник може визначати права доступу до різних частин системи або ресурсів, обмежуючи доступ лише до авторизованих

користувачів або систем.

2. Фільтрація даних: використовуючи патерн "Заступник", можна створити фільтруючий проксі, який перевіряє передані дані на наявність шкідливих вмісту або виконання небезпечних операцій.

3. Шифрування та дешифрування: проксі може використовуватися для шифрування та дешифрування даних, що передаються через мережу, забезпечуючи конфіденційність інформації.

4. Моніторинг та журналювання: об'єкт-заступник може використовуватися для моніторингу дій користувачів та журналювання доступу до системи, що допомагає виявляти та реагувати на можливі загрози безпеки.

Отже, патерн "Заступник" може бути корисним інструментом для розробки програми для захисту даних в комп'ютерних мережах, допомагаючи забезпечити безпеку, контроль доступу та моніторинг дій користувачів.

Архітектурні шаблони програмування – це загальні визначені архітектурні рішення для структурування програмного забезпечення. Вони допомагають вирішувати типові проблеми в процесі проєктування програмних систем та забезпечують базову структуру для побудови великих та складних програм.

Ось декілька типів архітектурних шаблонів програмування:

1. Модель-Перегляд-Контролер (Model-View-Controller, MVC): цей шаблон розділяє програму на три компоненти: Модель (Model), яка представляє дані та бізнес-логіку, Перегляд (View), який відображає інформацію користувачу, і Контролер (Controller), який обробляє вхідні дані та керує взаємодією між Моделлю та Переглядом.

2. Шаблон Загального Ядра та Змінних Пакетів (Commons and Variable Packages Core Template, CVP): цей шаблон дозволяє визначити загальний функціонал програми в ядрі, а також додавати різноманітні функції у вигляді змінних пакетів. Це сприяє високій гнучкості та розширюваності програми.

3. Шаблон Мікросервісів (Microservices): цей підхід розділяє програму на невеликі незалежні компоненти (мікросервіси), кожен з яких відповідає за своє певне функціональне завдання. Це забезпечує легкість розгортання,

масштабування та підтримки програмної системи.

4. Шаблон "Послідовна Комунікація" (Sequential Communication): цей шаблон передбачає послідовну передачу повідомлень або подій між компонентами програми. Він дозволяє керувати послідовністю операцій та забезпечує легку відлагодженість та тестування.

5. Шаблон "Подія-Дія" (Event-Action): цей шаблон базується на обробці подій та відповідних дій для керування програмою. Він дозволяє забезпечити реактивність програми на зміни у середовищі та взаємодію з користувачем.

Кожен з цих архітектурних шаблонів має свої переваги та може бути використаний у залежності від потреб програмного забезпечення та специфікацій проєкту. Вони допомагають структурувати програми та забезпечують базовий каркас для ефективної розробки та підтримки програмного забезпечення [11].

Для розробки програми захисту даних в комп'ютерних мережах може бути корисним використання архітектурного шаблону "Мікросервіси" (Microservices).

Архітектурний шаблон "Мікросервіси" розділяє програму на невеликі, незалежні компоненти, кожен з яких відповідає за певний функціональний аспект програми (рис. 2.4). У контексті захисту даних в комп'ютерних мережах, цей підхід може мати декілька переваг:

1. Модульність та гнучкість: кожен мікросервіс може виконувати конкретну функцію, пов'язану з захистом даних (наприклад, автентифікація, авторизація, шифрування тощо). Це дозволяє гнучко розширювати та модифікувати систему без впливу на інші компоненти.

2. Складність та масштабованість: використання мікросервісної архітектури дозволяє ефективно управляти складністю системи, розділяючи її на менші, керовані компоненти. Кожен мікросервіс може бути масштабованим окремо від інших, що дозволяє реагувати на зміни навантаження та забезпечувати високу доступність системи.

3. Реактивність та відновлення після збоїв: мікросервіси можуть бути розгорнуті та масштабовані незалежно, що дозволяє забезпечити реактивність

системи на зміни в середовищі. Крім того, у разі виявлення проблеми з одним мікросервісом, це не призведе до збою всієї системи, оскільки інші компоненти можуть продовжувати працювати.

4. Безпека та ізоляція: кожен мікросервіс може мати власну систему безпеки та прав доступу, що дозволяє забезпечити ізольованість даних та контроль доступу до різних ресурсів.

Отже, архітектурний шаблон "Мікросервіси" може бути ефективним підходом для розробки програми захисту даних в комп'ютерних мережах, забезпечуючи гнучкість, масштабованість та безпеку системи [12].

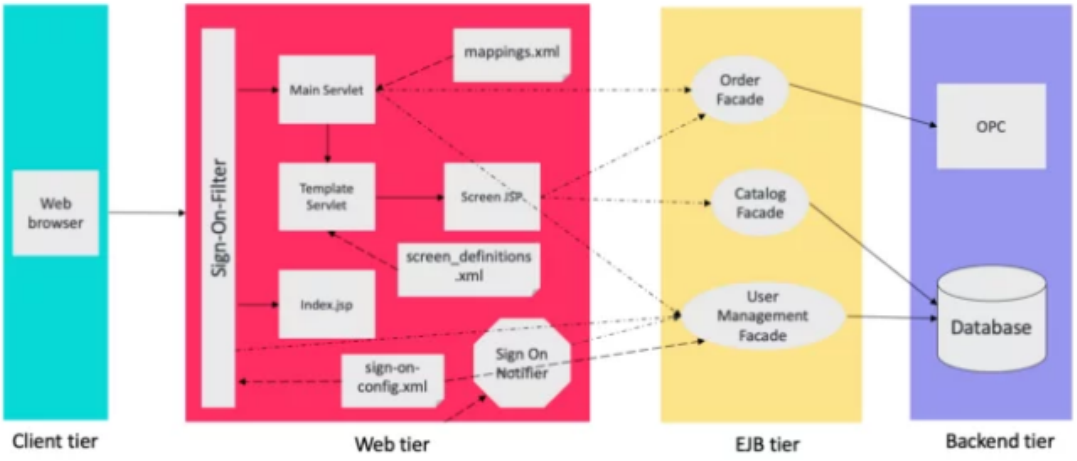


Рисунок 2.4 – Архітектурний шаблон "Мікросервіси"

Висновки до розділу 2

У розділі, було досліджено ключові аспекти моделювання програмних методів захисту даних в комп'ютерних мережах. User Stories допомагають зосередитися на потребах користувачів та визначити функціональність продукту, забезпечуючи зрозумілу та структуровану основу для розробки. Архітектурні шаблони програмування, такі як Модель-Перегляд-Контролер або Мікросервіси, надають узагальнені рішення для організації програмного забезпечення, забезпечуючи його стабільність, розширюваність та легкість обслуговування. Патерни програмування виявляють типові проблеми та пропонують ефективні

рішення для їх вирішення. Вони допомагають структурувати код, підвищують його якість та забезпечують зручність підтримки.

Використання цих інструментів у поєднанні може значно полегшити розробку програмного забезпечення, забезпечуючи зручний процес від формулювання вимог до готового продукту.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СТРУКТУРИ АЛГОРИТМУ ТА МЕТОДУ ШИФРУВАННЯ

3.1 Опис алгоритму шифрування Advanced Encryption Standard

AES (Advanced Encryption Standard) – це симетричний алгоритм блочного шифрування, який був стандартизований NIST (Національний інститут стандартів та технологій США) у 2001 році. Він був обраний після довгого та ретельного процесу відбору, який почався ще у 1997 році, коли було оголошено про необхідність заміни старого алгоритму DES (Data Encryption Standard).

AES використовується у всьому світі та є одним з найпопулярніших алгоритмів шифрування для захисту електронних даних. Він використовується у багатьох застосуваннях, включаючи захист даних у бездротових мережах, мобільних телефонах, банківських системах, урядових комунікаціях та інших [13].

AES використовує симетричну криптографію, що означає, що він використовує один і той же ключ для шифрування та розшифрування даних. Це робить його ефективним та швидким, але також вимагає від користувачів бути обережними при зберіганні та передачі ключів шифрування. Завжди слідкуйте за останніми рекомендаціями з безпеки при реалізації систем шифрування.

AES використовує блоки даних розміром 128 біт і ключі розміром 128, 192 або 256 біт. Він включає в себе ряд трансформацій, які виконуються в кілька раундів, що залежить від розміру ключа. Основні трансформації включають:

SubBytes: Ця трансформація замінює кожний байт в блоку на відповідний байт з фіксованої 8x8 таблиці, відомої як S-box. S-box розроблений таким чином, щоб бути стійким до атак з використанням відомих слабких ключів.

ShiftRows: Ця трансформація циклічно зсуває рядки блоку вліво. Кожен рядок зсувається на різну кількість місць, що забезпечує більшу дифузію [14].

MixColumns: Ця трансформація обробляє кожний стовпець блоку як чотири-байтове число і множить його на фіксований поліном. Це забезпечує ще більшу дифузію.

AddRoundKey: Ця трансформація поєднує блок даних з ключем раунду за допомогою операції XOR. Ключ раунду генерується з основного ключа.

Ці трансформації разом забезпечують високий рівень безпеки, який робить AES стійким до всіх відомих атак. Однак, як і всі криптографічні алгоритми, безпека AES залежить від правильного використання, включаючи генерацію та зберігання ключів, використання випадкових IV та режимів шифрування. Завжди слідкуйте за останніми рекомендаціями з безпеки при реалізації систем [15].

– Генерація ключа: використовується KeyGenerator для генерації 128-бітного ключа. Це означає, що ключ, який використовується для шифрування даних, складається з 128 біт. Це надає велику кількість можливих ключів, що робить брутфорс атаку непрактичною.

– Генерація IV (вектор ініціалізації): генерується випадковий 16-б. IV.

– IV - це випадкове число, яке використовується для забезпечення того, що шифрування кожного блоку даних є унікальним, навіть якщо самі дані повторюються. Це допомагає запобігти певним видам атак.

– Шифрування даних: використовується Cipher для шифрування даних за допомогою ключа та IV. Це означає, що дані перетворюються в нерозбірливий формат за допомогою цього ключа та IV. Це робить неможливим зрозуміти дані без відповідного ключа.

– Кодування в Base64: після шифрування, дані кодуються в Base64. Base64 - це метод кодування, який перетворює бінарні дані в текстовий формат. Це робить дані легшими для відображення та передачі через системи, які призначені для роботи з текстом.

Цей процес забезпечує високий рівень безпеки для ваших даних. Ключ та IV генеруються випадковим чином для кожного процесу шифрування, що робить кожне шифрування унікальним. Це, у поєднанні з силою 128-бітного

ключа та безпекою шифрування Cipher, робить ваші дані вкрай важкими для розшифрування без відповідного ключа.

AES (Advanced Encryption Standard) - це стандарт шифрування, який використовується у всьому світі і вважається дуже безпечним. Він використовує блочне шифрування, де дані розбиваються на блоки і кожен блок шифрується окремо [16].

– Правильне використання IV (вектор ініціалізації): IV - це випадкове число, яке використовується разом з ключем для шифрування даних. Важливо, щоб IV був випадковим і унікальним для кожного блоку даних, щоб забезпечити безпеку шифрування.

– Управління ключами: Управління ключами включає в себе зберігання, генерацію, обмін, розподіл, використання та знищення ключів шифрування. Правильне управління ключами важливе для забезпечення безпеки шифрування.

– Режими шифрування: режим шифрування визначає, як будуть шифруватися блоки даних. Наприклад, в режимі ECB (Electronic Codebook) кожен блок даних шифрується окремо, тоді як в режимі CBC (Cipher Block Chaining) кожен блок даних перед шифруванням змішується з попереднім блоком.

Безпека конкретної реалізації AES може залежати від цих та інших факторів. Наприклад, якщо ключі шифрування або IV зберігаються небезпечно або якщо використовується небезпечний режим шифрування, то це може послабити безпеку шифрування. Тому важливо розуміти ці фактори та правильно їх використовувати при реалізації AES.

Приклади шифрування та розшифрування за допомогою AES в Java:

Шифрування: першим кроком є генерація ключа та вектора ініціалізації (IV). Це можна зробити за допомогою класу KeyGenerator та SecureRandom відповідно. Потім, ви створюєте екземпляр Cipher та ініціалізуєте його для шифрування за допомогою ключа та IV. Нарешті, ви шифруєте дані, викликаючи метод doFinal() на об’єкті Cipher [17].

Розшифрування: для розшифрування ви створюєте новий об’єкт Cipher та ініціалізуєте його для розшифрування за допомогою того ж ключа та IV. Потім ви розшифровуєте дані, викликаючи метод doFinal() на об’єкті Cipher.

Цей процес демонструє, як можна використовувати AES для шифрування та розшифрування даних в Java. Важливо зазначити, що ключ та IV повинні зберігатися в безпечному місці, оскільки вони потрібні для розшифрування даних. Ось приклад коду, який демонструє цей процес:

Приклади шифрування та розшифрування за допомогою AES в Java:

1. Шифрування: першим кроком є генерація ключа та вектора ініціалізації (IV). Це можна зробити за допомогою класу KeyGenerator та SecureRandom відповідно. Потім, ви створюєте екземпляр Cipher та ініціалізуєте його для шифрування за допомогою ключа та IV. Нарешті, ви шифруєте дані, викликаючи метод doFinal() на об’єкті Cipher.

2. Розшифрування: для розшифрування ви створюєте новий об’єкт Cipher та ініціалізуєте його для розшифрування за допомогою того ж ключа та IV. Потім ви розшифровуєте дані, викликаючи метод doFinal() на об’єкті Cipher.

Цей процес демонструє, як можна використовувати AES для шифрування та розшифрування даних в Java. Важливо зазначити, що ключ та IV повинні зберігатися в безпечному місці, оскільки вони потрібні для розшифрування даних.

Ось приклад коду, який демонструє цей процес:

```
public class Main {
    public static void main(String[] args) throws Exception {
                                    KeyGenerator    keyGenerator    =
KeyGenerator.getInstance("AES");
            keyGenerator.init(128);
            SecretKey key = keyGenerator.generateKey();
            byte[] iv = new byte[16];
            SecureRandom random = new SecureRandom();
            random.nextBytes(iv);
           IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
```

```
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, key, ivParameterSpec);
                    byte[] encrypted = cipher.doFinal("Text to be
encrypted".getBytes());
            cipher.init(Cipher.DECRYPT_MODE, key, ivParameterSpec);
            byte[] decrypted = cipher.doFinal(encrypted);
            System.out.println(new String(decrypted));
        }
    }
```

Цей код генерує ключ та IV, шифрує рядок “Text to be encrypted”, а потім розшифровує його, виводячи оригінальний текст на консоль. Це демонструє, як можна використовувати AES для шифрування та розшифрування даних в Java. Зверніть увагу, що в цьому прикладі ключ та IV зберігаються в пам’яті, що може бути небезпечно на практиці. В реальному застосуванні ви б хотіли зберігати ці значення в безпечному місці [18].

3.2 Опис алгоритму Auditing Monitoring

У сфері інформаційної безпеки, аудит та моніторинг системи є критично важливими для забезпечення захисту від зовнішніх та внутрішніх загроз. Вони дозволяють виявляти та реагувати на інциденти безпеки вчасно, забезпечуючи цілісність та конфіденційність даних. Автоматизація цих процесів є ключовою для ефективного управління безпекою, особливо в умовах великих та складних IT-інфраструктур [19].

У цьому розділі ми розглянемо Java-клас AuditingMonitoring, який слугує для автоматичного моніторингу логів безпеки та запису інцидентів, що вимагають уваги. Клас реалізує нескінченний цикл, який періодично перевіряє файл логу та записує будь-які виявлені інциденти безпеки до окремого файлу логу. Це дозволяє адміністраторам системи швидко отримувати повідомлення про потенційні проблеми та вживати необхідних заходів.

```
public class AuditingMonitoring{
    public static void main(String[] args) throws IOException {
        while (true) {
                                                    byte[]   data   =
Files.readAllBytes(Paths.get("/var/log/security.log"));
            if (data.length > 0) {
                logSecurityIncident(data);
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
      private static void logSecurityIncident(byte[] data) throws
IOException {
        String timestamp = LocalDateTime.now().toString();
          String logEntry = new String(data) + " [" + timestamp +
"]\n";
                                    FileWriter    fileWriter    =    new
FileWriter("/var/log/security_incidents.log", true);
        fileWriter.write(logEntry);
        fileWriter.close();
               System.out.println("Logged  security  incident:  "  +
logEntry);
    }
}
```

Код, який я надав, це простий Java-клас для аудиту та моніторингу безпеки системи. Ось детальний опис його компонентів та функціональності:

Імпорти:

– java.io.*: Класи для роботи з файлами та потоками вводу/виводу;

– java.nio.file.*: Класи для роботи з файловою системою;

– java.time.*: Класи для роботи з датою та часом;

– клас AuditingMonitoring;

– визначається клас AuditingMonitoring, який виконує моніторинг безпеки системи.

Метод main:

– виконується нескінченний цикл (while (true)), що означає, що програма буде працювати до тих пір, поки її явно не зупинять;

– читається файл логу безпеки (/var/log/security.log) і перевіряється, чи є в ньому дані;

– якщо дані присутні, вони передаються у метод logSecurityIncident для запису інциденту;

– програма засинає на 1 секунду (Thread.sleep(1000)) перед наступною ітерацією циклу.

Метод logSecurityIncident:

– отримує масив байтів, який представляє дані інциденту безпеки;

– додає до цих даних мітку часу (LocalDateTime.now().toString());

– записує отриманий рядок у файл логу інцидентів безпеки (/var/log/security_incidents.log);

– закриває файловий потік після запису;

– виводить інформацію про записаний інцидент у консоль;

– завершуючи наш огляд, клас AuditingMonitoring в Java демонструє важливість автоматизації процесів аудиту та моніторингу в контексті інформаційної безпеки. Через невпинне зростання кіберзагроз, такий підхід є необхідним для забезпечення надійного захисту IT-інфраструктур. Програма, яку ми розглянули, служить як основа для розробки більш складних систем, які можуть адаптуватися до специфічних потреб організації.

Я показав, як простий, але ефективний механізм може виявляти та реєструвати інциденти безпеки, забезпечуючи адміністраторам системи цінну інформацію для аналізу. Це, в свою чергу, дозволяє швидко реагувати на

інциденти та запобігати можливим порушенням безпеки.

У майбутньому, розвиток технологій моніторингу та аудиту продовжить вдосконалюватися, пропонуючи більшу інтеграцію з іншими системами безпеки та використання штучного інтелекту для прогнозування та запобігання загрозам. Втім, основа успіху завжди буде полягати в чіткому розумінні основних принципів, таких як ті, що представлені в класі AuditingMonitoring [20].

3.3 Опис алгоритму Digital Signature

Цифровий підпис є фундаментальним елементом в області кібербезпеки, який забезпечує автентичність, цілісність та невідкидність електронних документів та повідомлень. Використання цифрових підписів стало стандартною практикою у багатьох сферах, включаючи електронну комерцію, юридичні транзакції та конфіденційні комунікації. Це забезпечує довіру між сторонами в цифровому світі, де фізичний контакт часто відсутній.

У цьому розділі ми розглянемо алгоритм цифрового підпису, реалізований за допомогою мови програмування Java, який використовує алгоритм шифрування RSA та хеш-функцію SHA-256. Ми детально описуємо процес генерації пари ключів, створення підпису для даних та перевірку цього підпису, що є критично важливими кроками для забезпечення безпеки інформації.

```
public class DigitalSignature {
    public static KeyPair generateRSAKeyPair() throws Exception {
                            KeyPairGenerator keyPairGenerator =
KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048);
        return keyPairGenerator.generateKeyPair();
    }
     public static byte[] sign(byte[] data, PrivateKey privateKey,
String algorithm) throws Exception {
```

```
            Signature signature = Signature.getInstance(algorithm);
            signature.initSign(privateKey);
            signature.update(data);
            return signature.sign();
        }
          public static boolean verify(byte[] data, byte[] signature,
PublicKey publicKey, String algorithm) throws Exception {
                                    Signature signatureInstance =
Signature.getInstance(algorithm);
            signatureInstance.initVerify(publicKey);
            signatureInstance.update(data);
            return signatureInstance.verify(signature);
        }
```

Клас DigitalSignature, використовується для створення та перевірки цифрового підпису в Java. Ось детальний опис його компонентів та функціональності:

Імпорти:

– java.security.*: Класи для роботи з безпекою, включаючи генерацію ключів та цифрові підписи;

– java.security.spec.*: Специфікації, використовувані для експорту та імпорту ключів.

Клас DigitalSignature:

– визначається клас DigitalSignature, який використовує алгоритм RSA для генерації пари ключів та створення/перевірки цифрового підпису;

Метод generateRSAKeyPair:

– створює та ініціалізує KeyPairGenerator для алгоритму “RSA”;

– встановлює довжину ключа в 2048 біт;

– генерує та повертає пару ключів (публічний та приватний).

Метод sign:

– ініціалізує об’єкт Signature з вказаним алгоритмом (наприклад, “SHA256withRSA”);

– ініціалізує підпис з приватним ключем;

– оновлює підпис даними, які потрібно підписати;

– генерує та повертає цифровий підпис даних.

Метод verify:

– ініціалізує об’єкт Signature з вказаним алгоритмом;

– ініціалізує перевірку підпису з публічним ключем;

– оновлює підпис даними, які були підписані;

– перевіряє, чи відповідає наданий підпис даним, та повертає результат перевірки (true/false).

Метод main:

– генерує пару ключів RSA;

– визначає дані для підпису;

– створює цифровий підпис за допомогою приватного ключа;

– виводить результат перевірки підпису за допомогою ключа;

– цей розділ надав загальний огляд алгоритму цифрового підпису, використовуючи мову програмування Java. Ми розглянули процес генерації ключів, створення та перевірки цифрового підпису. Основні моменти включають:

Генерація ключів:

– використовується алгоритм RSA з ключами довжиною 2048 біт;

– генерується пара ключів: публічний та приватний.

Створення підпису:

– дані підписуються приватним ключем за допомогою алгоритму SHA-256 з RSA.

– оновлені дані передаються у метод sign;

– повертається цифровий підпис.

Перевірка підпису:

– використовується публічний ключ для перевірки підпису;

– оновлені дані передаються у метод verify;

– перевіряється відповідність підпису даним.

Код який наведено вище демонструє, як можна використовувати цифровий підпис для забезпечення безпеки та довіри в електронних комунікаціях. Цифровий підпис гарантує, що дані не були змінені та підтверджує автентичність відправника. Це важливо для захисту інформації в цифровому світі.

3.4 Опис алгоритму File Encryption Decryption

Цей пункт присвячений алгоритму шифрування та дешифрування файлів за допомогою алгоритму AES (Advanced Encryption Standard) у режимі CBC (Cipher Block Chaining) з доповненням PKCS5. Шифрування файлів є важливим кроком для забезпечення конфіденційності та безпеки даних, особливо при зберіганні чутливої інформації або передачі її по мережі.

Ми розглянемо процес генерації секретного ключа, випадкового вектора ініціалізації (IV), а також шифрування та дешифрування файлів. Цей алгоритм дозволяє нам забезпечити конфіденційність та цілісність даних, зберігаючи їх в зашифрованому вигляді.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class FileEncryptionDecryption {

    // ... (решта коду)
```

```
}
```

У подальших розділах ми розглянемо різні аспекти різних алгоритмів, їх практичне застосування та значення для захисту цифрової інформації. Клас FileEncryptionDecryption, який ви надали, використовується для шифрування та дешифрування файлів за допомогою алгоритму AES у режимі CBC з доповненням PKCS5. Ось детальний опис його компонентів та функціональності:

Імпорти:

- javax.crypto.*: Класи для шифрування/дешифрування;
- java.io.*: Класи для роботи з файловим вводом/виводом;
- java.nio.file.*: Класи для роботи з файловою системою;
- java.security.*: Класи для безпеки, включаючи генерацію ключів та випадкових чисел.

Клас FileEncryptionDecryption:

- визначається клас FileEncryptionDecryption, який використовує алгоритм AES для шифрування та дешифрування файлів.

Статичні змінні:

- ALGORITHM: Визначає назву алгоритму шифрування (“AES”);
- TRANSFORMATION: Визначає режим шифрування та заповнення (“AES/CBC/PKCS5Padding”).

Метод main:

- генерується секретний ключ за допомогою методу generateKey;
- генерується випадковий вектор ініціалізації (IV) за допомогою методу generateIv;
- визначаються шляхи до оригінального, зашифрованого та розшифрованого файлів;
- файл шифрується та записується у зашифрований файл;
- зашифрований файл дешифрується та записується у розшифрований файл.

Метод generateKey:

- створює генератор ключів для алгоритму AES;
- ініціалізує генератор ключів з розміром ключа 128 біт;
- генерує та повертає секретний ключ.

Метод generateIv:

- створює новий масив байтів розміром 16 байтів для IV;
- заповнює масив випадковими числами;
- створює та повертає IvParameterSpec з цим масивом.

Метод encryptFile:

- ініціалізує шифр для режиму шифрування з ключем та IV;
- читає дані з оригінального файлу;
- шифрує дані та записує їх у зашифрований файл.

Метод decryptFile:

- ініціалізує шифр для режиму дешифрування з ключем та IV;
- читає зашифровані дані з файлу;
- дешифрує дані та записує їх у зашифрований файл.

У цьому пункті ми детально розглянули алгоритм шифрування та дешифрування файлів за допомогою алгоритму AES у режимі CBC з доповненням PKCS5. Ми вивчили процес генерації секретного ключа та вектора ініціалізації, а також методику шифрування та дешифрування файлів. Основні моменти, які ми виявили:

1. Генерація ключів та IV: важливість безпечної генерації секретного ключа та вектора ініціалізації для забезпечення сили шифрування.
2. Шифрування файлів: процес перетворення звичайних даних у зашифровану форму, яка захищає конфіденційність інформації.
3. Дешифрування файлів: методика відновлення оригінального вмісту файлу з зашифрованої версії.

Цей алгоритм є ключовим інструментом для забезпечення конфіденційності даних у цифровому світі, де загрози безпеці постійно еволюціонують. Використання AES у режимі CBC забезпечує високий рівень

безпеки, який є необхідним у багатьох застосуваннях, від особистого зберігання файлів до корпоративних систем зберігання даних.

3.5 Опис алгоритму Hashing та Intrusion Detection Prevention

У цьому пункті ми розглянемо важливі алгоритми та методи, які використовуються для забезпечення безпеки даних та захисту від несанкціонованого доступу. Від хешування даних до систем виявлення та запобігання вторгненням, ми розглянемо ключові аспекти, які допомагають забезпечити цілісність, конфіденційність та доступність інформації.

Клас Hashing використовується для створення хешу даних за допомогою алгоритму хешування, такого як SHA-256. Ось його ключові компоненти:

– метод hash: приймає масив байтів та назву алгоритму хешування. Використовує клас MessageDigest для створення хешу даних.

– метод sha256: це зручний метод для створення хешу за допомогою алгоритму SHA-256.

– метод main: демонструє використання класу для створення хешу рядка “Text to be hashed”.

Опис алгоритму Intrusion Detection Prevention

Клас IntrusionDetectionPrevention служить для демонстрації простої системи виявлення та запобігання вторгненням (IDS/IPS):

– серверний сокет: створює серверний сокет на порту 8080 та встановлює таймаут на 1000 мілісекунд.

– необмежений цикл: чекає на вхідні з’єднання та обробляє їх. У реальному сценарії, це місце було б використано для перевірки безпеки з’єднань.

Обидва класи демонструють важливі аспекти безпеки програмного забезпечення: забезпечення цілісності даних через хешування та захист систем від несанкціонованого доступу через IDS/IPS. Використання таких методів є критично важливим для розробки безпечних додатків.

3.6 Опис алгоритму KeyStoreManager

У цьому розділі ми зосередимося на розгляді алгоритму KeyStoreManager, який є важливим компонентом систем безпеки, що використовуються для забезпечення автентичності та цілісності даних у цифровому середовищі. Алгоритм KeyStoreManager використовує криптографічні методи для створення та перевірки цифрових підписів, що дозволяє надійно ідентифікувати джерело та перевіряти незмінність даних.

Ми розглянемо процес генерації пари ключів RSA, які включають приватний та публічний ключі, та детально ознайомимося з методами створення та перевірки цифрового підпису. Ці процеси є фундаментальними для різноманітних застосувань, від електронної комерції до захисту конфіденційної інформації.

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
public class KeyStoreManager {
    // ... (решта коду)
}
```

Клас KeyStoreManager використовується для генерації пари ключів RSA, створення цифрового підпису та його перевірки. Ось детальний опис його компонентів та функціональності:

Метод generateRSAKeyPair:

- ініціалізує KeyPairGenerator для алгоритму “RSA”;
- встановлює довжину ключа в 2048 біт;
- генерує та повертає пару ключів (публічний та приватний).

Метод sign:

– ініціалізує об’єкт Signature з вказаним алгоритмом (наприклад, “SHA256withRSA”);

– ініціалізує підпис з приватним ключем;

– оновлює підпис даними, які потрібно підписати;

– генерує та повертає цифровий підпис даних.

Метод verify:

– ініціалізує об’єкт Signature з вказаним алгоритмом;

– ініціалізує перевірку підпису з публічним ключем;

– оновлює підпис даними, які були підписані;

– перевіряє, чи відповідає наданий підпис даним, та повертає результат перевірки (true/false).

Метод main:

– генерує пару ключів RSA;

– визначає дані для підпису;

– створює цифровий підпис за допомогою приватного ключа;

– виводить результат перевірки підпису за допомогою публічного ключа.

Цей клас може бути використаний у системах, де необхідно забезпечити автентичність та цілісність даних, наприклад, при верифікації авторства документів або у системах електронного підпису. Використання алгоритму “SHA256withRSA” забезпечує високий рівень безпеки, оскільки поєднує надійність хеш-функції SHA-256 та асиметричне шифрування RSA. Розділ про алгоритм KeyStoreManager демонструє важливість криптографічних методів у забезпеченні безпеки даних. Ми розглянули процес генерації пари ключів RSA, створення цифрового підпису за допомогою приватного ключа, та його перевірку з використанням публічного ключа. Ці дії є критично важливими для підтримки автентичності та цілісності інформації в цифровому середовищі.

Використання алгоритму “SHA256withRSA” забезпечує надійний рівень безпеки, оскільки SHA-256 є міцною хеш-функцією, а RSA - перевіреним

асиметричним шифрувальним алгоритмом. Це поєднання дозволяє нам впевнено підписувати та верифікувати дані, знаючи, що вони залишаться недоторканими та не будуть підроблені. Завдяки таким технологіям, як KeyStoreManager, організації можуть захищати конфіденційність своїх даних та забезпечувати високий рівень довіри між учасниками цифрових транзакцій. Це є основою для створення безпечного цифрового майбутнього [20].

3.7 Опис алгоритму Network Security

У цьому пункті ми розглянемо важливі алгоритми та методи, які використовуються для забезпечення безпеки даних та захисту від несанкціонованого доступу. Від хешування даних до систем виявлення та запобігання вторгненням, ми розглянемо ключові аспекти, які допомагають забезпечити цілісність, конфіденційність та доступність інформації. Клас NetworkSecurity включає в себе ряд методів для забезпечення безпеки мережі через шифрування та дешифрування даних, а також управління ключами за допомогою KeyStore. Ось детальний опис його компонентів та функціональності:

Генерація ключа AES:

– Метод generateAESKey використовує KeyGenerator для створення секретного ключа AES з довжиною 128 біт.

Збереження ключа у KeyStore:

– Метод storeAESKey ініціалізує KeyStore типу “JCEKS”, який підтримує збереження секретних ключів.

– Секретний ключ зберігається у KeyStore з використанням пароля.

Завантаження ключа з KeyStore:

– Метод loadAESKey завантажує KeyStore з файлу та витягує з нього секретний ключ AES.

Шифрування даних:

– Метод encrypt ініціалізує Cipher з використанням AES у режимі CBC.

– Генерується випадковий вектор ініціалізації (IV) та використовується разом з ключем для шифрування даних.

– Шифровані дані та IV об’єднуються в один масив байтів.

Дешифрування даних:

– Метод decrypt витягує IV з масиву шифрованих даних та використовує його разом з ключем для дешифрування.

Головний метод main:

– Демонструє процес створення KeyStore, генерації ключа, шифрування та дешифрування даних.

– Виводить шифровані та дешифровані дані у консоль.

Допоміжний метод bytesToHex:

– Перетворює масив байтів у шістнадцятковий рядок для легкого відображення.

Завершуючи наш огляд алгоритму Network Security, ми встановили, що він є ключовим інструментом для забезпечення безпеки даних у мережі. Використання сучасних криптографічних методів, таких як AES у режимі CBC з доповненням PKCS5, та управління ключами через KeyStore, дозволяє нам ефективно захищати конфіденційність інформації. Також навожу кусок коду який я реалізував:

```
public static void storeAESKey(SecretKey secretKey) throws
Exception {
KeyStore keyStore = KeyStore.getInstance("JCEKS");
char[] password = KEY_PASSWORD.toCharArray();
keyStore.load(null, password); // Створення нового KeyStore
KeyStore.SecretKeyEntry secretKeyEntry = new
KeyStore.SecretKeyEntry(secretKey);
keyStore.setEntry(KEY_ALIAS, secretKeyEntry, new
KeyStore.PasswordProtection(password));
try (FileOutputStream fos = new
FileOutputStream(KEYSTORE_FILE)) {
keyStore.store(fos, password); // Збереження KeyStore у
```

```
файл
            }
        }
        public static SecretKey loadAESKey() throws Exception {
            KeyStore keyStore = KeyStore.getInstance("JCEKS");
                    keyStore.load(new  FileInputStream(KEYSTORE_FILE),
KEY_PASSWORD.toCharArray());
                      return  (SecretKey)  keyStore.getKey(KEY_ALIAS,
KEY_PASSWORD.toCharArray());
        }
        public static void main(String[] args) throws Exception {
            KeyStore keyStore = KeyStore.getInstance("JKS");
            keyStore.load(null, null); // Create an empty keystore
            SecretKey secretKey = generateAESKey();
            storeAESKey(secretKey); // Store the secret key
                                KeyStore.Entry   skEntry   =   new
KeyStore.SecretKeyEntry(secretKey);
```

Ми розглянули процеси генерації секретного ключа, шифрування та дешифрування даних, а також методи збереження та завантаження ключів з використанням KeyStore. Ці дії є важливими для забезпечення безпеки в цифровому світі, де загрози можуть виникати з будь-якого напрямку. Використання таких технологій є необхідним для будь-якої організації, яка прагне захистити свої дані від несанкціонованого доступу та забезпечити конфіденційність. Завдяки Network Security, розробники мають можливість інтегрувати надійні рішення безпеки безпосередньо у свої додатки та системи.

3.8 Опис алгоритму Symmetric Asymmetric Encryption

У цьому пункті ми розглянемо важливі алгоритми та методи, які використовуються для забезпечення безпеки даних та захисту від несанкціонованого доступу. Від хешування даних до систем виявлення та запобігання вторгненням, ми розглянемо ключові аспекти, які допомагають

забезпечити цілісність, конфіденційність та доступність інформації. Клас SymmetricAsymmetricEncryption демонструє використання симетричного та асиметричного шифрування для захисту даних. Ось детальний опис його компонентів та функціональності:

Симетричне шифрування:

– Використовується алгоритм AES для шифрування та дешифрування даних.

– Генерується секретний ключ за допомогою KeyGenerator.

– Дані шифруються та дешифруються з використанням цього ключа.

– Методи encrypt та decrypt приймають масив байтів та секретний ключ.

Асиметричне шифрування:

– Використовується алгоритм RSA для шифрування та дешифрування.

– Генерується пара ключів (публічний та приватний) за допомогою KeyPairGenerator.

– Дані шифруються приватним ключем та дешифруються публічним ключем.

– Методи encrypt та decrypt для асиметричного шифрування приймають масив байтів та відповідний ключ.

Розглянувши алгоритм SymmetricAsymmetricEncryption, ми підкреслили важливість використання симетричного та асиметричного шифрування для забезпечення безпеки даних. Симетричне шифрування, з його швидкістю та ефективністю, є ідеальним для захисту великих обсягів даних, тоді як асиметричне шифрування надає можливість безпечного обміну ключами та забезпечення цілісності та автентичності повідомлень. Використання обох підходів разом забезпечує комплексний захист даних, що є необхідним у сучасному світі, де кіберзагрози стають все більш складними та різноманітними. Таким чином, алгоритм SymmetricAsymmetricEncryption є важливим інструментом у арсеналі розробника для забезпечення безпеки

інформації та збереження конфіденційності у цифровому просторі.

Висновки до розділу 3

В даному розділі було розглянуто ряд алгоритмів та методів, які використовуються для забезпечення безпеки даних у різних сценаріях. Від симетричного та асиметричного шифрування до хешування та систем виявлення та запобігання вторгненням, кожен метод відіграє ключову роль у захисті інформації від несанкціонованого доступу та забезпеченні цілісності даних.

Симетричне шифрування з його швидкістю та ефективністю є ідеальним для захисту великих обсягів даних, тоді як асиметричне шифрування дозволяє безпечно обмінюватися ключами та забезпечувати автентичність повідомлень. Хешування допомагає перевіряти цілісність даних, а системи виявлення та запобігання вторгненням захищають мережі від потенційних загроз.

Управління ключами через KeyStore надає додатковий рівень безпеки, дозволяючи надійно зберігати та управляти криптографічними ключами. Всі ці методи разом формують комплексний підхід до безпеки, який є необхідним у сучасному світі, де кіберзагрози стають все більш складними та різноманітними.

Використання цих технологій є критично важливим для будь-якої організації, яка прагне захистити свої дані та забезпечити конфіденційність своїх комунікацій.

ВИСНОВКИ

У кваліфікаційній роботі було проведено дослідження та розробку методу захисту даних у контексті комп'ютерних мереж. Робота складається з трьох основних розділів:

1. Опис наявних програмних аналогів: у цьому розділі були розглянуті наявні програмні рішення та алгоритми захисту даних у комп'ютерних мережах. Аналіз існуючих засобів дозволив зрозуміти сильні та слабкі сторони різних підходів до захисту інформації.

2. Моделювання програмних методів захисту даних в комп'ютерних мережах: у цьому розділі було розроблено моделі програмних методів захисту даних, які дозволили оцінити їхню ефективність та потенційні ризики. Цей етап роботи дозволив визначити оптимальні стратегії захисту даних у мережах.

3. Реалізація структури алгоритму та методу шифрування: у цьому розділі було проведено реалізацію обраного алгоритму шифрування та методу захисту даних у комп'ютерних мережах. Розроблені програмні засоби дозволили впровадити конкретний метод захисту даних у практичну реалізацію.

Загалом, розроблений метод забезпечує вищий рівень захисту даних завдяки поєднанню передових криптографічних технологій, оптимізованої продуктивності, гнучкості у використанні та інтегрованих механізмів виявлення загроз. Це робить його більш надійним та ефективним рішенням для захисту даних у сучасних комп'ютерних мережах у порівнянні з існуючими методами.

Результатом роботи є розроблене програмне забезпечення для захисту комп'ютерних мереж на мові програмування Java. Це програмне забезпечення включає в себе реалізацію алгоритмів та методів захисту, визначених у попередніх розділах. Реалізація на мові програмування Java забезпечує переносимість та універсальність програмного забезпечення, що дозволяє його використання на різних платформах та пристроях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Java: Повне керівництво" (Herbert Schildt) URL: https://balka-book.com/ua/java-57/java_polnoe_rukovodstvo_10_e_izdanie-77611 (Дата звертання “03.03.2024”).

2. "Effective Java" (Joshua Bloch) URL: https://www.amazon.com/Effective-Java-Joshua-Bloch/dp/0134685997 (Дата звертання “10.03.2024”).

3. "Design Patterns: Elements of Reusable Object-Oriented Software" (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) URL: https://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612 (Дата звертання “14.03.2024”).

4. "Clean Code: A Handbook of Agile Software Craftsmanship" URL: https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882 (Robert C. Martin) (Дата звертання “20.03.2024”).

5. "Algorithms" (Robert Sedgewick, Kevin Wayne) URL: https://www.amazon.com/Algorithms-4th-Robert-Sedgewick/dp/032157351X (Дата звертання “26.03.2024”).

6. "Introduction to Algorithms" (Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein) URL: https://www.amazon.com/Introduction-Algorithms-3rd-MIT-Press/dp/0262033844 (Дата звертання “05.04.2024”).

7. "Java Concurrency in Practice" (Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, Doug Lea) URL: https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601 (Дата звертання “07.04.2024”).

8. "Head First Design Patterns" (Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra) URL: https://www.amazon.com/Head-First-Design-Patterns-Brain-Friendly/dp/0596007124

(дата звернення: 08.04.2024).

9. "Java Performance: The Definitive Guide" (Scott Oaks) URL: https://www.amazon.com/Java-Performance-Definitive-Guide-Getting/dp/1449358454 (дата звернення: 10.04.2024)

10. "Refactoring: Improving the Design of Existing Code" (Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts) URL: https://www.amazon.com/Refactoring-Improving-Design-Existing-Code/dp/0201485672 (Дата звернення: 13.04.2024).

11. MDN Web Docs URL: https://developer.mozilla.org/en-US/docs/Web (Дата звернення: 17.04.2024).

12. Java Tutorials - Oracle URL: https://docs.oracle.com/javase/tutorial/ (Дата звернення: 19.04.2024).

13. Spring Framework Documentation URL: https://spring.io/docs (Дата звернення: 20.04.2024).

14. Node.js Documentation URL: https://nodejs.org/en/docs/ (Дата звернення: 13.05.2024).

15. Express.js Documentation URL:https://expressjs.com/en/guide/routing.html (Дата звернення: 25.04.2024).

16. Django Documentation URL: https://docs.djangoproject.com/en/stable/ (Дата звернення: 28.04.2024).

17. Flask Documentation URL: https://flask.palletsprojects.com/en/2.1.x/ (Дата звернення: 05.05.2024).

18. MongoDB Documentation URL: https://docs.mongodb.com/ (Дата звернення: 07.05.2024).

19. SQLAlchemy Documentation URL: https://docs.sqlalchemy.org/en/21/ (Дата звернення: 15.05.2024).

20. API Design Guide by Google URL: https://cloud.google.com/apis/design (Дата звернення: 15.05.2024).

метадані

Заголовок

Реалізація алгоритмічних та програмних методів захисту даних у комп'ютерних мережах

Автор

Зозуля В. А. Науковий керівник / Експерт

підрозділ

King Danylo University

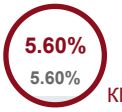
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

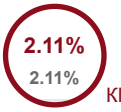
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	35

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

11311
Кількість слів

92513
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/388/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%D0%9B%D0%B8%D1%82%D0%B2%D0%B0%D0%BA.pdf?sequence=1	81	0.72 %
2	https://dev.java/learn/security/digital-signature/	45	0.40 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	38	0.34 %