Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ВЕБ-ДОДАТОК ДЛЯ АНАЛІЗУ КРЕДИТНИХ ПРОПОЗИЦІЙ ФІНАНСОВОГО РИНКУ

WEB APPLICATION FOR ANALYSIS OF CREDIT OFFERS OF THE FINANCIAL MARKET

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Конон Андрій Миколайович

(підпис)

Керівник:
к.т.н., доцент
Мельник Катерина Вікторівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» червня 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____

«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Конону Андрію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Веб-додаток для аналізу кредитних пропозицій фінансового ринку*

Керівник роботи *к.т.н., доцент Мельник Катерина Вікторівна*

затверджені наказом закладу вищої освіти від «28» грудня 2022 року № 982/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2023р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Огляд існуючого веб-застосунку для аналізу кредитних пропозицій фінансового ринку та інструментів моніторингу продуктивності
Моніторинг продуктивності інформаційної системи та її рефакторинг
Забезбечення безпеки інформаційної системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Існуючі рішення
Використані технології SolarWinds, Datadog, NewRelic, AppDynamics, Splunk
Схема роботи програмного продукту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Мельник К.В.*		
Теоретичне дослідження та практична реалізація	*Мельник К.В.*		
Практична реалізація об'єкта проектування	*Мельник К.В.*		
Висновки			

7. Дата видачі завдання 01.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Обґрунтування теми*	До 15.11.2022 р.	
2.	*Огляд літератури із досліджуваної проблеми*	До 15.12.2022 р.	
3.	*Розділ 1*	До 02.02.2023 р.	
4.	*Розділ 2*	До 02.03.2023 р.	
5.	*Висновки та пропозиції*	До 02.04.2023 р.	
6.	*Формування списку використаних джерел*	До 15.04.2023 р.	
7.	*Формування додатків*	До 02.05.2023 р.	
8.	*Оформлення ілюстративного матеріалу*	До 15.05.2023 р.	
9.	*Нормоконтроль*	До 25.05.2023 р.	
10.	*Інструментальна перевірка на академічний плагіат*	До 01.06.2023 р.	
11..	*Представлення кваліфікаційної роботи бакалавра до захисту*	До 07.06.2023 р.	

Здобувач вищої освіти _____ Конон А.М.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Мельник К.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Конон А.М. Веб-додаток для аналізу кредитних пропозицій фінансового ринку. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2023.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, трьох додатків.

Перший розділ присвячено огляду існуючого веб-застосунку для аналізу кредитних пропозицій фінансового ринку та інструментів моніторингу продуктивності.

В другому розділі здійснено моніторинг продуктивності інформаційної системи та її рефакторинг.

Третій розділ присвячено опису забезпечення безпеки інформаційної системи: проблематика безпеки інформаційних систем, огляд інструментів для сканування додатку щодо можливих вразливостей, вибір інструменту для сканування веб-застосунку щодо можливої загрози безпеки, використання інструменту Snyk для усунення можливої загрози безпеки, аналіз результатів використання Snyk.

Об'єкт – веб-застосунок для аналізу кредитних пропозицій фінансового ринку. Предмет – моніторинг продуктивності та відстеження вразливостей безпеки веб-застосунку.

Метою роботи є вибір інструментів для моніторингу продуктивності додатку та відстеження вразливостей безпеки застосунку та його залежностей. Інтеграція та використання вибраних інструментів для зменшення ризиків безпеки та покращення продуктивності додатку.

Ключові слова: продуктивність, моніторинг, програмне забезпечення, швидкість виконання, рефакторинг, безпека даних, інструменти, програмний код, вразливість, база даних.

ANNOTATION

Konon A.M. A web application for analyzing financial market credit offers. Manuscript.

Bachelor's qualifying thesis of the OP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2023.

The qualification work consists of an introduction, three sections, conclusions, a list of used sources, and three appendices.

The first section is dedicated to the overview of the existing web application for analyzing financial market credit offers and performance monitoring tools.

In the second section, the monitoring of the information system performance and its refactoring was carried out.

The third section is devoted to the description of information system security: issues of information system security, review of tools for scanning the application for possible vulnerabilities, selection of a tool for scanning a web application for a possible security threat, use of the Snyk tool to eliminate a possible security threat, analysis of the results of using Snyk.

The object is a web application for analyzing financial market credit offers.

The subject is performance monitoring and tracking web application security vulnerabilities.

The purpose of the work is the selection of tools for monitoring the performance of the application and tracking the security vulnerabilities of the application and its dependencies. Integrate and use selected tools to reduce security risks and improve application performance.

Keywords: monitoring, vulnerabilities, web application security.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Фінансовий ринок пропонує величезну кількість кредитних продуктів (споживчі кредити, іпотека, автокредити, кредитні картки) від банків, мікрофінансових організацій (МФО) та фінтех-компаній. Пересічному користувачеві складно порівняти всі пропозиції, правильно оцінити повну вартість кредиту (APR/EAPR) з урахуванням прихованих комісій, страховок та різних схем погашення. Це створює попит на незалежні, прозорі та автоматизовані інструменти порівняння. Але особливу увагу слід приділяти аналізу технічного стану інформаційних систем (ІС), що є критичною для будь-якої сучасної організації, оскільки ІС становлять фундамент бізнес-процесів, прийняття рішень та конкурентоспроможності. Такий аналіз, часто відомий як ІТ-аудит, дозволяє оцінити, наскільки ефективно, надійно та безпечно функціонують апаратне забезпечення, програмне забезпечення, мережева інфраструктура та відповідні процедури.

Аналіз допомагає ідентифікувати компоненти (сервери, мережеве обладнання, бази даних), які працюють на межі своїх можливостей або мають високий ризик збою. Своєчасне виявлення та усунення технічних проблем (наприклад, перегрів обладнання, застаріле програмне забезпечення, неефективні конфігурації) запобігає критичним простоям бізнес-процесів, що безпосередньо призводять до фінансових втрат. Технічний аналіз включає ідентифікацію слабких місць в інфраструктурі, наприклад, некоректно налаштовані міжмережеві екрани, неліцензійне ПЗ, застарілі прошивки, які можуть бути використані для кібератак. Оцінка цілісності та безпеки систем зберігання даних є важливою для захисту конфіденційної інформації клієнтів і компанії. Якщо компанія планує зростання, тобто збільшення кількості користувачів, обсягів даних, впровадження нових продуктів, то аналіз стану ІС показує, чи здатна поточна інфраструктура підтримати ці плани без значних збоїв.

Отже, аналіз технічного стану інформаційних систем об’єднує в собі: аналіз апаратного та програмного забезпечення, мережевої інфраструктури, резервне копіювання, відновлення після збою (Disaster Recovery), моніторинг, управління доступом.

Об’єктом роботи є веб-додаток для аналізу кредитних пропозицій фінансового ринку.

Предметом є моніторинг продуктивності та відстеження вразливостей безпеки веб-додатку.

Метою роботи є вибір інструментів для моніторингу продуктивності програми та відстеження вразливостей програми та її залежностей. Інтеграція та використання вибраних інструментів для зниження ризику безпеки та підвищення продуктивності програми.

На основі поставленої мети необхідно виконати наступні завдання:

– проаналізувати застосунок та його залежностей на рахунок можливих вразливостей безпеки;

– здійснити мінімізацію ризиків втрати даних користувачів та компанії;

– проаналізувати швидкості виконання програмного коду додатку;

– рефакторинг та зменшення часу виконання підпрограм інформаційної системи.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ВЕБ-ЗАСТОСУНКІВ ТА ІНСТРУМЕНТІВ МОНІТОРИНГУ ПРОДУКТИВНОСТІ

1.1 Аналіз фінансових веб-застосунків

Протягом останнього десятиліття відбулися значні зміни в традиційній банківській галузі, особливо в сферах платежів, кредитування, управління активами та роздрібного банкінгу. Сучасний фінансовий сектор значною мірою перейшов у формат веб-застосунків (web applications), які пропонують послуги від простого банкінгу до складного інвестиційного аналізу.

Проводячи огляд основних існуючих фінансових веб-застосунків на світовому та, частково, українському ринках, можна виділити наступні категорії:

– цифровий банкінг та необанки: веб-платформи, які надають повний спектр банківських послуг через інтернет, часто з мінімальною фізичною присутністю або взагалі без неї;

– PFM-застосунки, які допомагають користувачам відстежувати, аналізувати та планувати свої витрати й доходи, часто агрегуючи дані з різних банківських рахунків;

– інвестиції та трейдинг: платформи, що надають доступ до фінансових ринків (акції, облігації, ETF, криптовалюти);

– кредитування та аналіз пропозицій: ці платформи спрощують процес отримання кредиту та порівняння умов;

– платіжні системи та міжнародні перекази: фокус на швидких, безпечних та часто недорогих переказах коштів між країнами або між фізичними/юридичними особами.

До цифрового банкінга та необанків відносять: традиційний онлайн-банкінг (веб-версії систем для керування поточними рахунками, здійснення платежів, відкриття депозитів, наприклад, Приват24, Ощад 24/7, портали великих міжнародних банків); необанки (платформи, що працюють виключно через цифрові канали, фокусуються на зручності, низьких комісіях та

інноваційних UX-рішеннях наприклад, Revolut, Monobank, Starling Bank.); корпоративний банкінг (застосунки для юридичних осіб: зарплатні проєкти, міжнародні платежі, керування фінансами підприємства, наприклад, портали для бізнес-клієнтів, Wise for Business).

PFM-застосунки виконують збір даних із різних джерел (банки, картки) в єдиний інтерфейс, а також автоматичну категоризацію транзакцій, встановлення лімітів, наприклад, Mint (Intuit), You Need A Budget (YNAB), Personal Capital (Empower). PFM-застосунки відстежують борги та виконання цілей: моніторинг кредитів та планування заощаджень на конкретні цілі (наприклад, нерухомість, відпустка). Прикладом є Credit Karma (фокус на кредитному рейтингу).

Веб-застосункі для отримання кредиту та порівняння умов мають різну функціональність:

– кредитні агрегатори: збір, порівняння та аналіз кредитних пропозицій (іпотека, споживчі кредити) від багатьох банків та МФО на єдиній платформі, наприклад, Finance.ua (рис.1.1), LUN.ua (для іпотеки), порівняльні сервіси в ЄС та США.

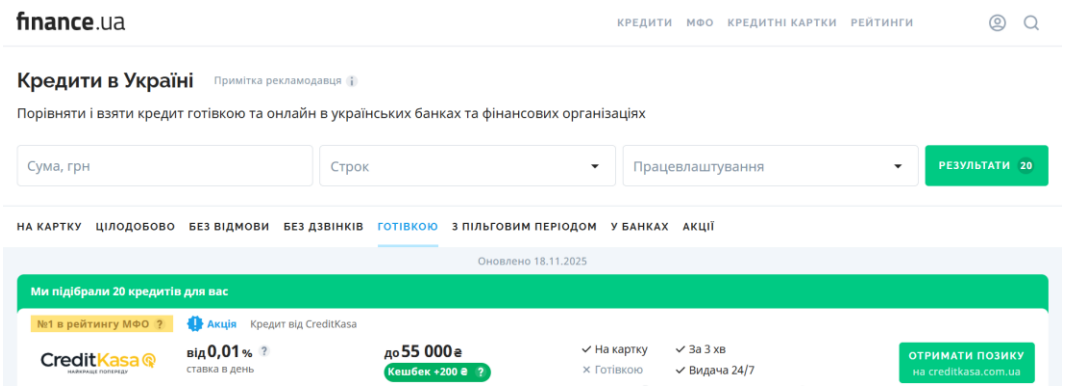

Рисунок 1.1 – Веб-застосунок Finance.ua

– P2P-кредитування (Peer-to-Peer): платформи, які напряму з'єднують приватних інвесторів та позичальників, минаючи банки, наприклад, LendingClub, Prosper.

– Оцінка кредитоспроможності: сервіси, що дозволяють користувачу перевірити свій кредитний рейтинг та отримати персоналізовані пропозиції, наприклад, Credit Karma.

Усі існуючі фінансові веб-застосунки активно розвиваються у напрямках AI та Machine Learning для прогнозування витрат, виявлення шахрайства та надання високоперсоналізованих фінансових порад; безпеці: багатофакторна автентифікація (MFA), біометрія, шифрування кінця в кінець; Open Banking / API: інтеграція між різними застосунками та банками для безшовного обміну фінансовими даними (за згодою користувача).

У зв’язку з цим компанії надають велике значення постійному вдосконаленню своїх інформаційних систем, оптимізації обчислень, інформаційній безпеці, перевірці програми на можливі загрози. На сьогоднішній день цей процес частково можна автоматизувати за допомогою спеціальних інструментів та сервісів

1.2 Огляд інструментів для аналізу технічного стану

Як правило використовується комплекс різноманітних методик, які можна згрупувати за об'єктом аналізу та способом збору даних, що забезпечує комплексну оцінку ефективності, безпеки та надійності ІС.

До основних методик відносять методи аудиту та документування, тобто проводится перевірка встановленим стандартам та фактична інвентаризація ресурсів. При інвентаризації ІТ-активів важливо виявити неліцензійного ПЗ та обладнання, термін експлуатації якого вийшов, провести порівняння фактичних налаштувань (серверів, мережевих пристроїв, баз даних) із рекомендованими стандартами або внутрішніми політиками безпеки. А також здійснити оцінку відповідності вимогам:

– GDPR (General Data Protection Regulation – загальний регламент про захист даних, закон Європейського Союзу, який регулює захист персональних даних громадян ЄС та ЄЕЗ. Він встановлює правила щодо збору, обробки та

зберігання персональних даних, надаючи фізичним особам контроль над інформацією про себе),

– PCI DSS – міжнародний стандарт безпеки даних індустрії платіжних карток, який складається з 12 вимог щодо захисту даних власників банківських карток, що зберігаються, обробляються або передаються організаціями).

– ISO/IEC 27001 – цеміжнародний стандарт, який встановлює вимоги до системи менеджменту інформаційної безпеки (ЗМІБ) організацій. Він допомагає компаніям управляти ризиками, захищаючи конфіденційність, цілісність та доступність інформації, що підвищує довіру клієнтів та партнерів.

До основних методик також відносять методики оцінки продуктивності та надійності. Для моніторингу продуктивності використовується інструмент APM (Application Performance Management) та системи моніторингу для постійного відстеження ключових метрик: завантаження CPU, використання пам'яті, пропускна здатність мережі, час відгуку додатків, швидкість роботи бази даних. Проводять штучне створення високого навантаження на систему для імітації пікових умов або DoS-атак, щоб визначити межі її стійкості та точку відмови. Для пошуку помилок, попереджень, спроб несанкціонованого доступу та аномалій проводять ручний або автоматизований (через системи SIEM або Log Aggregators) аналіз журналів подій.

Головна мета APM – перетворити реактивне управління (виправлення збоїв, коли вони вже сталися) на проактивне (виявлення та усунення проблем до того, як вони вплинуть на користувачів). Приклади популярних іструменти APM:

– Dynatrace – нтелектуальна платформа для моніторингу та аналітики, яка допомагає керувати продуктивністю додатків і хмарної інфраструктури. Вона використовує штучний інтелект (Davis AI) для автоматизації виявлення та усунення проблем, надаючи комплексні відповіді замість просто даних про продуктивність, безпеку додатків та цифровий досвід користувачів.

– New Relic – це хмарний сервіс для повного моніторингу та аналітики продуктивності програмного забезпечення (SaaS), який допомагає командам відстежувати веб- та мобільні додатки, а також інфраструктуру в режимі

реального часу. Він надає інструменти для візуалізації даних, аналізу помилок, моніторингу бази даних, а також для діагностики та усунення несправностей. .

– AppDynamics (Cisco) – корпоративна платформа для моніторингу продуктивності додатків (APM) і IT-аналітики, яка допомагає компаніям відстежувати та оптимізувати роботу своїх програм та інфраструктури. Вона дозволяє в реальному часі виявляти проблеми продуктивності, аналізувати їх причини та забезпечувати кращий користувацький досвід, а також інтегрується з іншими інструментами для автоматизації реагування на інциденти.

– Prometheus та Grafana: хоча це не повноцінний APM-пакет, ця комбінація часто використовується для збору та візуалізації метрик продуктивності в Open Source-рішеннях.

– Datadog: Об'єднана платформа для моніторингу хмарної інфраструктури, APM та логів.

Для оцінки безпеки використовують спеціалізоване програмне забезпечення (наприклад, Nessus, OpenVAS), що дозволяє автоматично здійснювати пошук відомих вразливостей у мережевому обладнанні, операційних системах та додатках. При тестуванні на проникнення імітують дії реального зловмисника, використовуючи виявлені вразливості, щоб спробувати отримати несанкціонований доступ до системи чи даних. А також здійснюють оцінку логічної структури ІС, зон довіри, розташування міжмережевих екранів (Firewalls), систем виявлення вторгнень (IDS/IPS) та механізмів сегментації мережі.

До основних методик також відносять методики оцінки операційної ефективності. Важливо фактично перевірити чи працюють процедури резервного копіювання та чи можна швидко і без втрат відновити критичні дані та системи після збою.

Для якісного аналізу технічного стану ІС зазвичай використовується комбінація розглянутих методик, що дозволяє отримати як статистичні дані (звітність, інвентаризація), так і динамічну оцінку (продуктивність, надійність, безпека).

Виходячи з цих критеріїв, необхідно вибрати інструмент моніторингу продуктивності програми, який підходить для нашої організації відповідно до її потреб і фінансових можливостей. Розглянемо наявний на даний момент APM.

SolarWinds Server – це не єдиний продукт, а скоріше загальна назва для комплексу інструментів від компанії SolarWinds, призначених для моніторингу, управління та аналізу продуктивності серверів, додатків та IT-інфраструктури. Найбільш відомим і центральним продуктом у цьому комплексі є SolarWinds Server & Application Monitor (SAM) [6].

Він проводить моніторинг продуктивності серверів, тобто відстеження основних метрик апаратного забезпечення та операційної системи (ОС) у режимі реального часу (рис.1.2):

– завантаження процесора (CPU)
– використання оперативної пам'яті (RAM)
– дисковий простір та I/O
– температура та стан обладнання (для фізичних серверів)

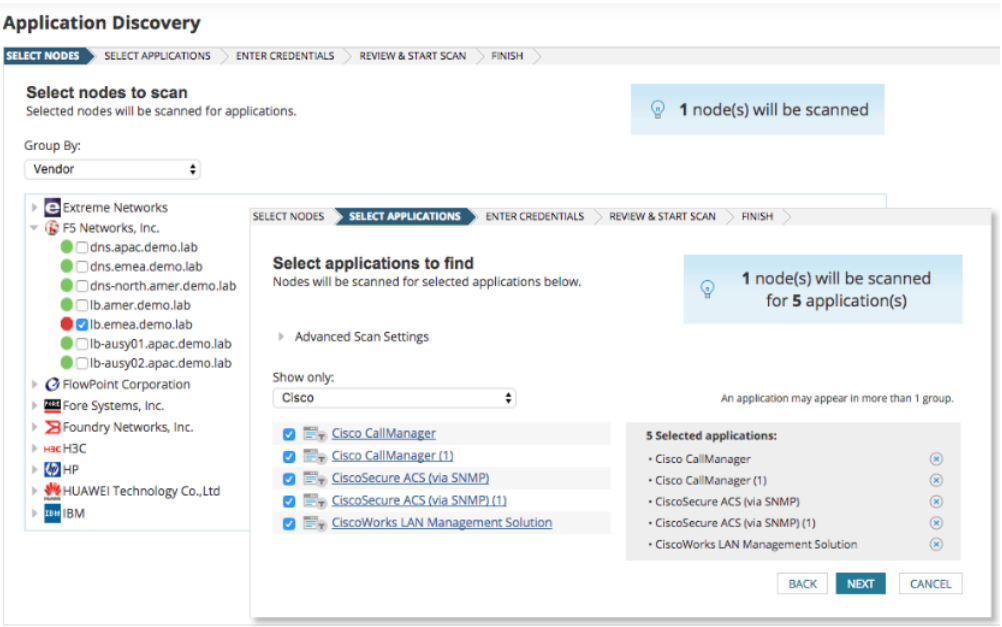

Рисунок 1.2 – Комплексний моніторинг серверних застосунків

Для оптимізованого використання ресурсів необхідно планування потужності сервера. Контроль потужності серверів, щоб планувати майбутнє

зростання. SolarWinds Observability Self-Hosted відстежує використання процесора, пам'яті та сховища, забезпечуючи вашим додаткам необхідні ресурси. Точне планування потужності серверів допомагає запобігти вузьким місцям і надає дієву аналітику для забезпечення безперебійної роботи ваших серверів і додатків (рис.1.3).

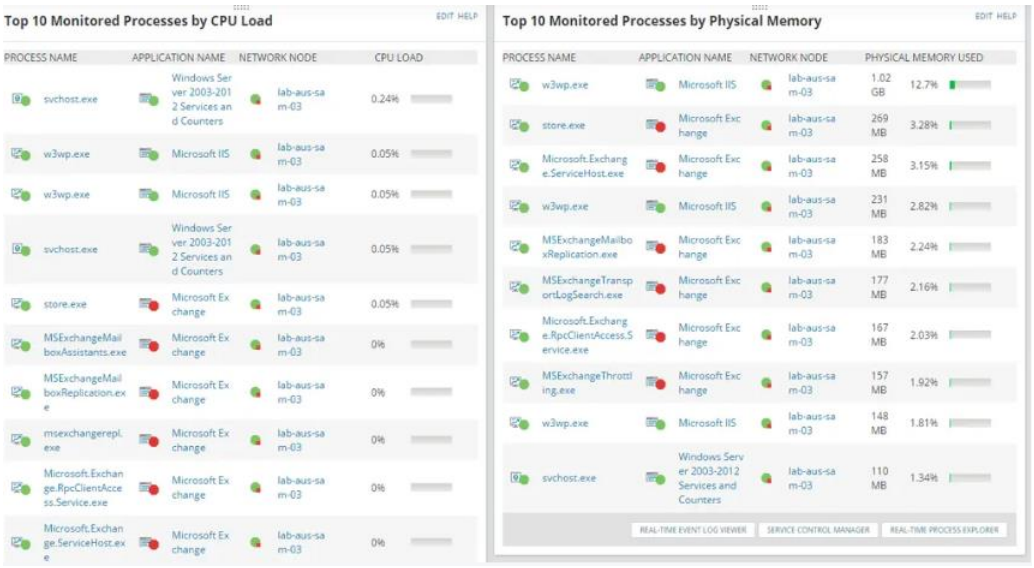

Рисунок 1.3 – Планування потужності сервера

SolarWinds Server and Application Monitor (SAM) проводить моніторинг застосунків. SAM містить готові шаблони для моніторингу популярного ПЗ:

– Microsoft Exchange та Active Directory;
– бази даних (SQL Server, Oracle, MySQL) (рис.1.4);
– веб-сервери (IIS, Apache);
– різні віртуальні машини та хмарні служби.

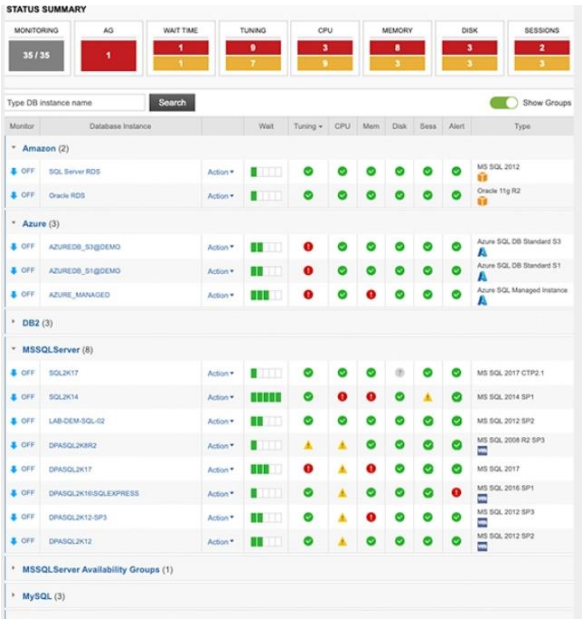

Рисунок 1.4 –Аналітика баз даних

Управління активами та інвентаризація включає збір детальних даних про програмне (рис.1.5) та апаратне забезпечення, встановлене на серверах (версії ОС, встановлені патчі, ліцензії).

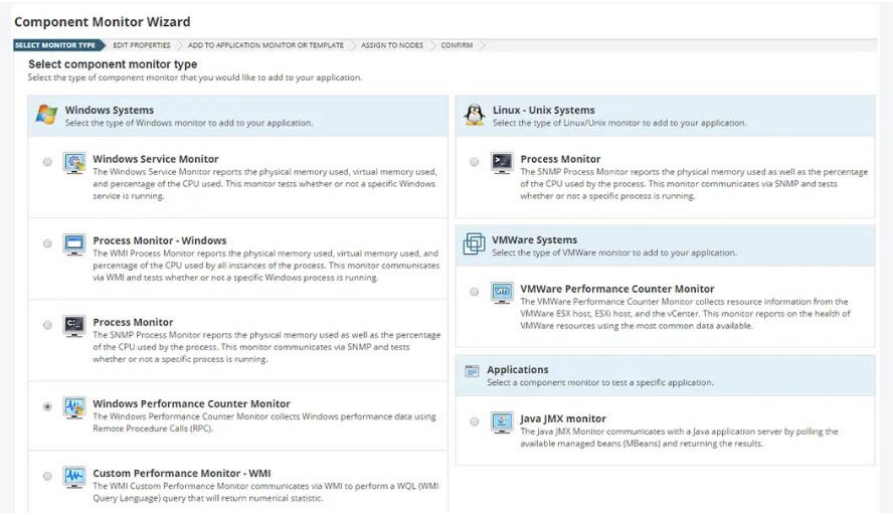

Рисунок 1.5 – Поглиблений моніторинг користувацьких програм

Можна налаштувати умови, при яких система автоматично надсилає сповіщення (наприклад, якщо завантаження CPU перевищує 90% протягом 5 хвилин) або автоматично виконує дії.

SolarWinds Server and Application Monitor (SAM) дозволяє керувати хмарною інфраструктурою (моніторинг середовища Azure, рис.1.6), такою як локальні сервери, відстежувати продуктивність, доступність та залежності, щоб забезпечити безперебійну роботу.

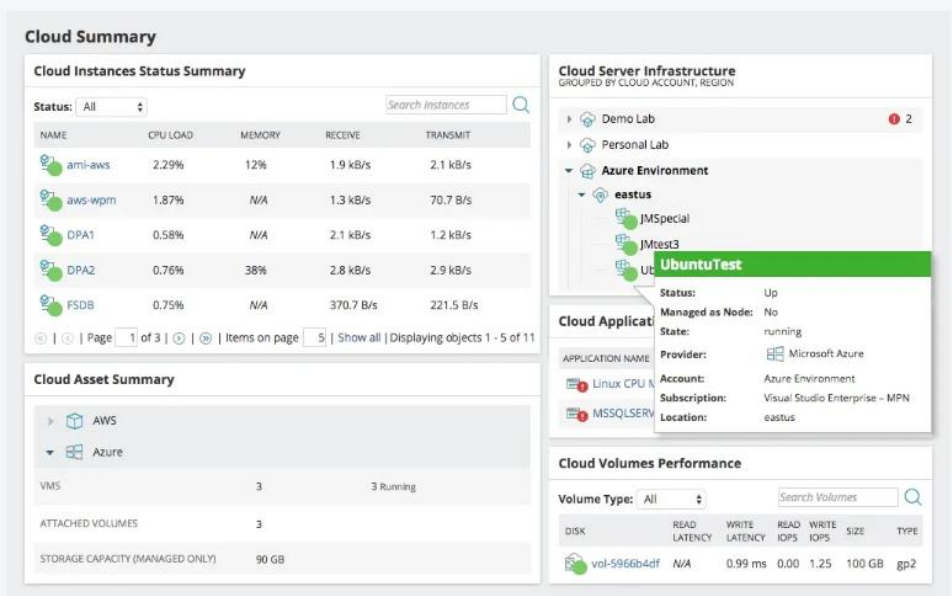

Рисунок 1.6 – Моніторинг середовища Azure

APM може аналізувати продуктивність та виконання коду, написаного на семи найпоширеніших мовах веб-додатків: Java, Node.js, PHP, .NET, Python, Ruby та Go.

SolarWinds часто пропонує інтегровані рішення, які доповнюють моніторинг серверів. В таблиці 1.1 наведено основні продукти SolarWinds.

Таблиця 1.1 – Релевантні продукти SolarWinds

Продукт	Сфера застосування
Orion Platform	Єдина інтегрована платформа, яка об'єднує дані з усіх модулів SolarWinds. SAM працює на цій платформі.
Network Performance Monitor (NPM)	Моніторинг та аналіз мережевого обладнання (маршрутизатори, комутатори) та пропускної здатності мережі.
Virtualization Manager (VMAN)	Моніторинг та управління віртуальними середовищами (VMware vSphere та Microsoft Hyper-V).
Database Performance Analyzer (DPA)	Глибокий аналіз продуктивності баз даних, виявлення повільних запитів.

Database Performance Analyzer (DPA) – це спеціалізований інструмент для глибокого аналізу та моніторингу продуктивності систем керування базами даних (СКБД) у режимі реального часу та історично.

Головна мета DPA полягає не лише у відстеженні загальних метрик (як-от завантаження CPU або використання пам'яті), а й у визначенні та детальному аналізі «вузьких місць» на рівні самої бази даних, особливо тих, що стосуються часу очікування (блокування, I/O-операції, очікування на доступ до пам'яті, повільні мережеві запити тощо).

DPA фіксує, скільки часу займає виконання кожного запиту, і класифікує цей час за типами очікування. Це дозволяє швидко ідентифікувати, чи викликана затримка неефективним кодом (повільний SQL-запит), конкуренцією ресурсів (блокування), чи проблемами з інфраструктурою (повільний диск). DPA також проводить діагностику SQL-запитів, тобто автоматичну ідентифікацію найвитратніших SQL-запитів, процедур та функцій, які найбільше навантажують СКБД та може запропонувати оптимальні індекси або зміну структури запиту, якщо СКБД вибрала неефективний шлях для отримання даних.

DPA зберігає історичні дані про продуктивність, що дозволяє адміністраторам виявляти тренди зниження продуктивності, порівнювати продуктивність до і після внесення змін (наприклад, оновлення коду, застосування патча).

DPA підтримує моніторинг найбільш поширених СКБД у єдиному інтерфейсі, таких наприклад, як Microsoft SQL Server, Oracle, MySQL та MariaDB, PostgreSQL, Azure SQL Database, Amazon RDS, Aurora

Splunk є інструментом моніторингу, який використовує штучний інтелект (AI) у своєму програмному забезпеченні. Це потужна, широко використовувана платформа для збору, індексації, пошуку та аналізу машиногенерованих даних (machine data), таких як логи (logs), метрики та події з додатків, серверів, мережевого обладнання та хмарних сервісів [8].

По суті, Splunk перетворює величезні обсяги неструктурованих даних на зрозумілу, доступну для пошуку та аналізу інформацію, що є критично важливим

для IT-операцій, безпеки та бізнес-аналітики. Отже, Splunk збирає дані з тисяч джерел у режимі реального часу, використовуючи спеціальні агенти (Forwarders) або прямі з'єднання, неструктуровані дані логів індексуються та зберігаються у сховищі. Користувачі використовують мова пошуку Splunk (SPL – Search Processing Language) для виконання складних запитів, кореляції подій з різних джерел та створення візуалізацій.

Splunk використовується для діагностики проблем, моніторінгу продуктивності, а також для аналізу трендів та метрик виявлення потенційних проблем (наприклад, зростання помилок або навантаження) до того, як вони призведуть до збою.

У сфері кібербезпеки Splunk функціонує як потужний інструмент SIEM (Security Information and Event Management), що дозволяє виявити загрози, відслідковувати відповідність нормативним вимогам (PCI DSS, GDPR, HIPAA) та розлідувати інциденти.

Splunk може аналізувати дані логів, щоб отримати інформацію про поведінку клієнтів та бізнес-процеси. Проводить аналіз часу виконання критично важливих для бізнесу транзакцій, наприклад, оформлення замовлення.

Splunk має ряд переваг серед платформ для операційної аналітики та управління безпекою, це масштабованість, унікальну мову запитів, яка дозволяє не лише шукати, але й агрегувати, корелювати, трансформувати та візуалізувати (рис.1.7) дані та має широку бібліотеку готових додатків та інтеграцій (Splunkbase) для роботи з AWS, Microsoft, Cisco та багатьма іншими вендорами.

Рисунок 1.7 – Візуалізація даних та звітність у Splunk

1.3 Використання інструменту Prometheus та Grafana

Prometheus та Grafana – це два взаємодоповнюючих інструментів з відкритим кодом, які складають де-факто стандартний стек для моніторингу та візуалізації метрик у сучасних хмарних та мікросервісних інфраструктурах. Разом вони утворюють потужну, гнучку та економічно ефективну систему спостережуваності.

Prometheus – це спеціалізована система моніторингу та оповіщення, яка була розроблена в SoundCloud і зараз є незалежним проєктом Cloud Native Computing Foundation (CNCF) (рис.1.8).

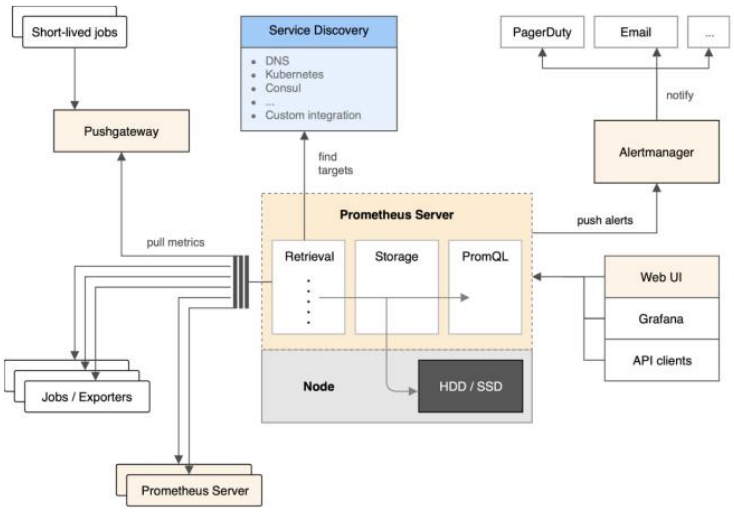

Рисунок 1.8 – Огляд архітектури Prometheus

На відміну від багатьох старих систем, Prometheus зазвичай працює за моделлю «pull». Він активно опитує цільові об'єкти (застосунки, сервери, сервіси) через HTTP-інтерфейс, щоб забрати їхні метрики. Prometheus зберігає всі дані як часові ряди, де кожна точка даних має значення і мітку часу. Кожен часовий ряд ідентифікується унікальним ім'ям метрики та набором labels. Prometheus має власну потужну мову запитів, яка дозволяє виконувати складні агрегації, розрахунки та аналіз метрик у режимі реального часу.

Prometheus інтегрується з широким спектром сервісів через експортери (Exporters) – це спеціалізований програмний компонент, який розгортається поряд із моніторинговою програмою. Його основна функція полягає у зборі необхідних показників із цільового застосунку (наприклад, MySQL, Node.js, Linux), їх конвертації у формат, зрозумілий для Prometheus, та обслуговуванні HTTP-запитів від сервера Prometheus. Фактично, експортер виступає як проксі-сервер, який уніфікує метричний інтерфейс програми під стандарт експозиції Prometheus.

Grafana – це провідний інструмент для візуалізації, аналізу та створення інтерактивних дашбордів. Хоча Grafana може працювати з багатьма джерелами даних (Prometheus, InfluxDB, PostgreSQL, Splunk, ElasticSearch тощо), вона є найбільш популярною саме як фронтенд для Prometheus. Вона дозволяє користувачам створювати настроювані інформаційні панелі (дашборди) з різними типами візуалізації: графіки, таблиці, лічильники, гістограми та використовує PromQL для формування запитів до Prometheus, дозволяючи відображати результати складних обчислень. Хоча Prometheus має власну систему сповіщень, Grafana також надає можливості налаштування сповіщень на основі даних, отриманих з джерел (з певної версії). Дозволяє створювати динамічні дашборди, де користувачі можуть легко обирати об'єкти моніторингу (наприклад, конкретний сервер чи мікросервіс) через випадаючі списки, не змінюючи запити (рис.1.9).

Рисунок 1.9 – Приклад панелі відображення візуалізованих даних в Grafana

Стек Prometheus та Grafana є класичним прикладом поділу функцій, де Prometheus виступає як бек-енд – збирає, зберігає, обробляє та створює сповіщення на основі метрик, а Grafana виступає як фронт-енд – запитує оброблені дані з Prometheus через PromQL і представляє їх у зручному, візуальному форматі.

Такий поділ ролей дозволяє кожному інструменту зосереджуватися на своєму основному завданні, забезпечуючи ефективність та гнучкість усієї системи моніторингу.

Висновок до розділу 1

Використання Prometheus та Grafana для моніторингу технічного стану інформаційної системи та виявлення слабких місць для подальшого вдосконалення є хорошим вибором. Prometheus збирає високодеталізовані метрики з дуже високою частотою (зазвичай кожні 15-30 секунд), що дозволяє фіксувати короткочасні піки навантаження та аномалії, які можуть бути «слабкими місцями».

Потужна мова запитів PromQL дозволяє створювати складні агрегації, розраховувати відсотки (p95, p99), середні значення та тренди. Це дає можливість точно ідентифікувати, наприклад, що лише 5% транзакцій виконуються повільно, і сфокусуватися на їх оптимізації.

Завдяки Експортерам, метрики збираються максимально близько до цільової програми (база даних, веб-сервер, додаток), що забезпечує точність даних.

РОЗДІЛ 2
МОНІТОРИНГ ТА РЕФАКТОРИНГ У ІНФОРМАЦІЙНІХ СИСТЕМАХ

2.1 Рефакторинг за допомогою Prometheus та Grafana

Рефакторинг – це процес структурного вдосконалення існуючого коду без зміни його функціональності. Головна мета зробити код чистішим, простішим, легшим для читання та підтримки, що в кінцевому підсумку підвищує його ефективність і гнучкість для подальшого розвитку. Рефакторинг є постійною, неперервною діяльністю в процесі розробки програмного забезпечення, а не одноразовою акцією. Отже, найважливіший принцип після рефакторингу програма має працювати точно так само, як і до нього, з точки зору користувача чи інших систем. Якщо функціональність змінилася, це вже не рефакторинг, а зміна вимог або впровадження нової функції.

Рефакторинг вирішує наступні проблеми: виділення повторюваних фрагментів у окремі функції чи класи; перейменування змінних, методів та класів, щоб вони точніше відображали свій зміст і призначення; розбиття великих і складних функцій на менші, більш керовані частини, що відповідають принципу єдиної відповідальності; оптимізація взаємозв'язків між об'єктами та модулями.

До 1990-х років рефакторинг був лише неформальною практикою. Однак докторські дисертації Вільяма Грізволда (1991) і Вільяма Опдайка (1992) стали першими науковими роботами, що систематизували цей процес. Вони описали не тільки загальну процедуру рефакторингу, а й створили визнаний каталог загальних методів, включаючи покрокові інструкції та чіткі індикатори для їх застосування, що стало основою для подальшого розвитку методології.

Використання стеку Prometheus та Grafana є часто незамінним для рефакторингу коду, оскільки воно забезпечує об'єктивні дані для вимірювання успіху чи невдачі внесених змін. Процес рефакторингу, підкріплений цими інструментами, перетворюється з суб'єктивного вдосконалення на керований даними процес оптимізації.

Стек виконує дві критично важливі функції під час рефакторингу: підтвердження необхідності та верифікація результату.

На першому етапі відбувається обґрунтування та ідентифікація слабких місць (до рефакторингу), тобто потрібно точно знати, які ділянки коду є найпроблемнішими. Prometheus збирає метрики за допомогою вбудованих у код експортерів або клієнтських бібліотек. Ці метрики показують: які функції або методи виконуються найдовше (latency); де найбільше споживається CPU чи пам'яті; які транзакції чи API-виклики генерують найбільше помилок.

Grafana візуалізує ці дані на дашбордах, дозволяючи команді точно визначити (наприклад, графік показує, що функція calculate_credit_score() є причиною 90% затримок). Це перетворює припущення («цей код виглядає погано») на факт, керований даними.

На другому етапі відбувається верифікація та оцінка ефекту. Після внесення змін (рефакторингу) необхідно негайно виміряти їхній вплив на продуктивність системи. Графіки в Grafana дозволяють порівняти ключові метрики (наприклад, час відгуку, завантаження CPU) до і після розгортання рефакторингового коду.

Для оцінка успіху:

– якщо час виконання критичної функції зменшився, рефакторинг успішний;

– якщо кількість помилок не зросла (або зменшилася), рефакторинг не спричинив нових регресій.

– якщо використання пам'яті знизилося, покращилася ефективність коду.

Іноді рефакторинг покращує одну метрику, але погіршує іншу (наприклад, код став швидшим, але почав споживати більше пам'яті). Стек Prometheus та Grafana дозволяє миттєво виявити такі несподівані побічні ефекти (регресії).

2.2 Мова заптів PromQL

Незалежно від того, чи відстежується використання процесора в кластері Kubernetes, чи моніториться затримка API , оволодіння PromQL (Prometheus Query Language) перетворює необроблені дані на практичні висновки.

Prometheus використовує власну потужну мову запитів, яка називається PromQL. PromQL дозволяє не просто витягувати необроблені дані, а й виконувати складні операції, агрегації, розрахунки та аналіз часових рядів.

PromQL працює з чотирма основними типами даних:

– миттєвий вектор (instant vector): набір часових рядів, кожен з яких містить одне значення для останнього моменту часу;

– діапазонний вектор (range vector): набір часових рядів, кожен з яких містить діапазон значень протягом заданого проміжку часу;

– просте числове значення;

– рядок (String).

Кожен стек моніторингу містить базові запити такий, як наприклад, простий вибір метрики: http_requests_total. Цей запит повертає всі часові ряди з назвою метрики http_requests_total.

Фільтрація за допомогою міток: http_requests_total{status="500"}. Цей запит фільтрує http_requests_total метрику, щоб відображати лише запити, які призвели до коду помилки 500.

Агрегування показників за міткою: sum(rate(http_requests_total[5m])) by (status). Цей запит підсумовує кількість усіх HTTP-запитів за 5 хвилин, але розділяє їх за кодом стану. Отримується чітке уявлення про рівень помилок у порівнянні з успішними запитами – ідеально підходить для виявлення проблем.

Пошук найкращих споживачів:

topk(5, sum(rate(node_cpu_seconds_total{mode!="idle"}[5m])) by (instance)).

Цей запит показує 5 екземплярів з найбільшим використанням процесора. Функція topk() ранжує результати, що дозволяє легко ідентифікувати тих, хто споживає більше ресурсів.

Розрахунок часу безвідмовної роботи сервісу:

(sum(up{job="api-server"}) / count(up{job="api-server"})) * 100

Цей запит надає відсоток API-серверів, які наразі працюють. Він ділить кількість «працюючих» екземплярів на загальну кількість і множить на 100. Простий, але неймовірно корисний для відстеження SLO.

predict_linear(node_filesystem_free_bytes{mountpoint="/"}[6h], 4 * 3600)

Цей запит прогнозує, скільки місця на диску буде через 4 години, на основі моделі використання за останні 6 годин. Ця predict_linear() функція виявляє проблеми до того, як вони виникнуть.

2.3 Prometheus для виявлення аномалій

Зі зростанням обсягів даних у Prometheus збільшується і «інформаційний шум», що ускладнює виявлення справжніх проблем. Традиційний підхід, який сьогодні є загальноприйнятим, покладається на людський фактор: адміністратори вручну встановлюють фіксовані порогові значення для графіків та сповіщень на інформаційних панелях.

Підхід на основі штучного інтелекту (ШІ) пропонує ефективніше рішення. Він передбачає навчання моделей машинного навчання на історичних даних часових рядів для точного прогнозування очікуваних метричних значень. Аномалія фіксується тоді, коли фактичне значення метрики суттєво відхиляється від прогнозованого моделлю. Це дозволяє автоматично виявляти нетипову поведінку системи.

Основні вимоги до системи виявлення аномалії Prometheus:

1) Prometheus метрики – важливі показники, які потрібно контролювати;

2) Grafana – інструмент візуалізації, призначений для створення графіків даних часових рядів Prometheus.

На рисунку 2.1 наведена приклад схеми поєднання цих двох інструментів в одну систему:

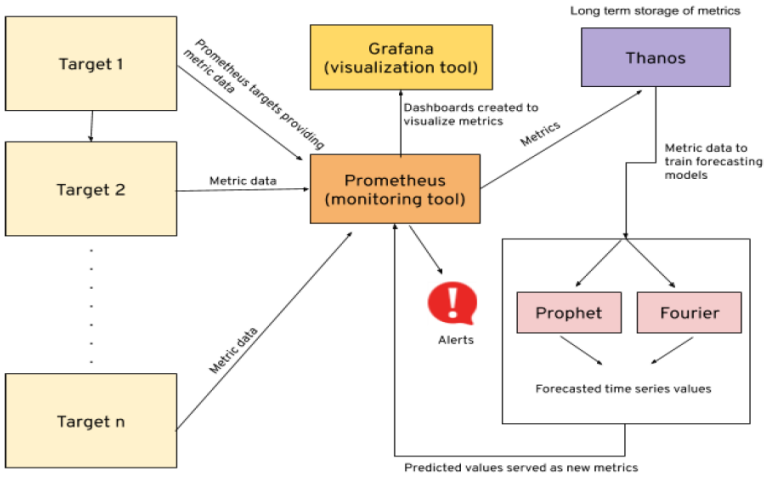

Рисунок 2.1 – Приклад архітектури системи виявлення аномалій на базі Prometheus та Grafana

2.4 Виявлення аномалій для GitLab

Однією з найбазовіших функцій мови запитів Prometheus є агрегація даних часових рядів у реальному часі. Ендрю Ньюдігейт, видатний інженер команди інфраструктури GitLab, висунув гіпотезу, що мову запитів Prometheus також можна використовувати для виявлення аномалій у даних часових рядів.

GitLab – це універсальна платформа для DevOps, яка надає інструменти для повного життєвого циклу розробки програмного забезпечення. Вона об’єднує в собі систему керування репозиторіями коду (на базі Git), інструменти для безперервної інтеграції та доставки (CI/CD), а також можливості для управління проєктами, відстеження помилок та спільної роботи команд.

Існує чотири основні причини, що визначають важливість виявлення аномалій для GitLab:

1. Діагностика інцидентів: виявити, які послуги вийшли за свої звичайні рамки, зменшивши час виявлення інциденту (MTTD), і, відповідно, запропонувати швидше вирішення проблеми.

2. Виявлення зниження продуктивності: наприклад, якщо до сервісу внесена регресія, що призводить до того, що він занадто часто звертається до іншого сервісу, можливість швидко виявити та усунути цю проблему.

3. Виявлення та усунення зловживань: GitLab надає механізми доставки та інтеграції (GitLab CI/CD) та хостингу (GitLab Pages) та обмеження кількість користувачів, які можуть цими механізмами користуватися.

4. Безпека: виявлення аномалій важливе для виявлення незвичайних тенденцій у тимчасових рядах GitLab.

По-перше, дані часових рядів мають бути правильно агреговані. Джерелом даних для демонстрації використовується стандартний лічильник *http_requests_total.* Цей приклад метрики має деякі додаткові виміри: *method, controller*, *status_code*, *environment,* а також виміри, які додає Prometheus, такі як *instance* та *job* (лістинг 2.1).

Лістинг 2.1 – Стандартний лічильник *http_requests_ total*

```
http_requests_total{
 job="apiserver",
 method="GET",
 controller="ProjectsController",
 status_code="200",
 environment="prod"
}
```

Кінець лістингу 2.1

Далі потрібно вибрати правильний рівень агрегації для даних, занадто велика агрегація даних може призвести до замалої кількості вимірів, що створить дві потенційні проблеми: можемо пропустити справжні аномалії, оскільки агрегація приховує проблеми, що виникають у підмножинах даних або виявиться аномалія, яку важко віднести до певної частини системи без подальшого дослідження аномалії.

Але якщо агрегувати дані занадто мало, отримається серія даних з дуже малими розмірами вибірки, що може призвести до хибнопозитивних результатів і означати позначення справжніх даних як викидів.

Правильним рівнем агрегації є рівень обслуговування, тому включаємо мітку *job* та мітку *environment*, але опускаємо інші виміри. Агрегація даних

включає: *job* http requests, швидкість за п'ять хвилин, що, по суті, є швидкістю для *job* та *environment* протягом п'ятихвилинного вікна (лістинг 2.2).

Лістинг 2.2 – Включення мітки *job* та мітки *environment*

```
- record: job:http_requests:rate5m
expr: sum without(instance, method, controller, status_code)
(rate(http_requests_total[5m]))
# --> job:http_requests:rate5m{job="apiserver", environment="prod"}  21321
# --> job:http_requests:rate5m{job="gitserver", environment="prod"}  2212
# --> job:http_requests:rate5m{job="webserver", environment="prod"}  53091
```

Кінець лістингу 2.2

Використання z-показника для виявлення аномалій.

Деякі основні принципи статистики можна застосувати для виявлення аномалій за допомогою Prometheus.

Якщо нам відоме середнє значення та стандартне відхилення (σ) ряду Prometheus, ми можемо використовувати будь-яку вибірку в ряду для обчислення z-score. *Z*-score вимірюється кількістю стандартних відхилень від середнього значення.

Отже, z-score 0 означатиме, що z-score ідентичний середньому значенню в наборі даних з нормальним розподілом, тоді як z-score 1 дорівнює 1,0 σ від середнього значення тощо.

Якщо припустити, що вихідні дані мають нормальний розподіл, 99,7% вибірок повинні мати z-score від нуля до трьох. Чим далі z-показник від нуля, тим менша ймовірність його існування. Ми застосовуємо цю властивість для виявлення аномалій у ряді Prometheus.

Обчислимо середнє значення та стандартне відхилення для метрики, використовуючи дані з великою вибіркою. У цьому прикладі використовується дані за один тиждень. Якщо припустити, що оцінюється правило запису раз на хвилину, то протягом тижня матимемо трохи більше 10 000 зразків (лістинг 2.3).

Лістинг 2.3 – Збирання даних раз на хвилину протягом тижня

```
# Long-term average value for the series
- record: job:http_requests:rate5m:avg_over_time_1w
expr: avg_over_time(job:http_requests:rate5m[1w])

# Long-term standard deviation for the series
- record: job:http_requests:rate5m:stddev_over_time_1w
expr: stddev_over_time(job:http_requests:rate5m[1w])
```

Кінець лістингу 2.3

Ми можемо обчислити *z*-score для запиту Prometheus, як тільки отримаємо середнє значення та стандартне відхилення для агрегації (лістинг 2.4).

Лістинг 2.4 – Обчислення *z*-score

```
# Z-Score for aggregation
(
job:http_requests:rate5m -
job:http_requests:rate5m:avg_over_time_1w
) /  job:http_requests:rate5m:stddev_over_time_1w
```

Кінець лістингу 2.4

Виходячи зі статистичних принципів нормального розподілу, можемо припустити, що будь-яке значення, яке виходить за межі діапазону приблизно від +3 до -3, є аномалією. Можемо створити сповіщення на основі цього принципу. Наприклад, можемо отримати сповіщення, коли наша агрегація знаходиться поза цим діапазоном більше п'яти хвилин.

RPS сервісу сторінок GitLab.com протягом 48 годин, з областю *z*-score ±3, виділеною зеленим (рис.2.2).

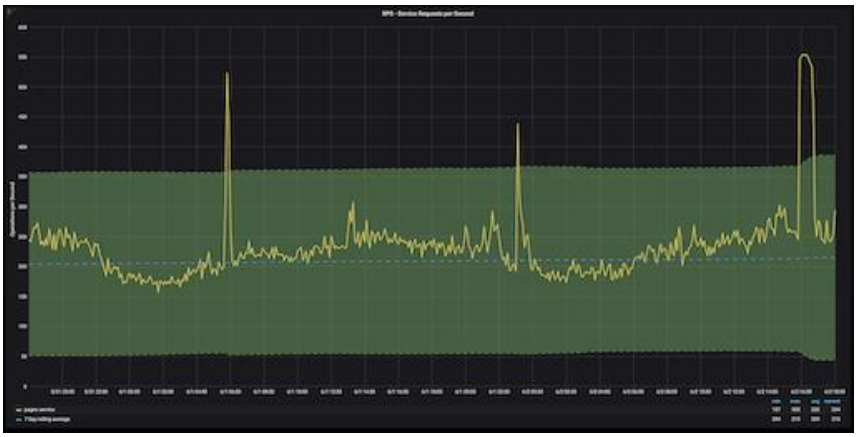

Рисунок 2.2 – Виявлення аномалій поза межами інтервалу від 3 до -3

Z-score дещо незручно інтерпретувати на графіку, оскільки вони не мають одиниці вимірювання. Але аномалії на цій діаграмі легко виявити. Все, що знаходиться поза зеленою зоною (яка позначає *z*-показники, що потрапляють у діапазон +3 або -3), є аномалією.

Висновок до розділу 2

У сучасній економіці цифрові платформи (веб-сайти та мобільні додатки) є ключовим, а часто й основним джерелом доходу компаній. Оскільки користувачі очікують бездоганного цифрового досвіду та розв'язання будь-яких проблем у режимі реального часу, проблеми з продуктивністю можуть серйозно підірвати зростання та прибутковість бізнесу.

Для підтримки розширення бізнесу та зростаючої кількості користувачів необхідно постійно збільшувати пропускну здатність інформаційної системи. Це досягається як шляхом розширення інфраструктури (збільшення кількості та потужності серверів), так і шляхом оптимізації програмного коду. Виявлення та прискорення повільних фрагментів коду за допомогою спеціальних аналітичних інструментів та метрик продуктивності є критично важливим для забезпечення високої ефективності системи.

Рішення класу Управління Продуктивністю Застосунків (APM) дозволяють компаніям не лише відстежувати відповідність ІС встановленим

стандартам, але й забезпечувати бездоганний досвід користувачів завдяки постійному моніторингу інфраструктури. Передові APM-системи надають розробникам глибоку аналітику, необхідну для проактивного виявлення та усунення проблем продуктивності до того, як вони вплинуть на кінцевого користувача, що безпосередньо сприяє підвищенню пропускної здатності та швидкості додатку.

На основі проведеного аналізу моніторингових інструментів, для подальшого використання було обрано стек Prometheus та Grafana. Завдяки цьому рішенню вдалося оптимізувати повільні запити до бази даних та здійснити успішний рефакторинг критичних фрагментів коду, що призвело до скорочення часу їх виконання.

Для гарантування швидкості та стабільності ІС, моніторинг і рефакторинг мають стати невіддільною та постійною частиною роботи інженерної команди. Хоча цей процес вимагає часових інвестицій, які могли б бути спрямовані на новий функціонал, у довгостроковій перспективі він забезпечує значно більшу цінність та прибутковість для бізнесу.

РОЗДІЛ 3
БЕЗПЕКА ВЕБ-ЗАСТОСУНКІВ

3.1 Проблематика безпеки

Проблематика безпеки веб-додатків має свою специфіку, оскільки веб-додаток є відкритою точкою доступу до внутрішньої інфраструктури компанії. Більшість атак спрямовані саме на програмне забезпечення а не лише на мережу. Основні проблеми безпеки веб-додатків можна згрупувати за наступними напрямками:

1. Вразливості коду. Це найбільш поширені та часто використовувані зловмисниками проблеми, які систематизовані, зокрема, у переліку OWASP Top 10.

– Ін'єкції. Найчастіше це SQL-ін'єкції, коли зловмисник впроваджує шкідливі SQL-команди через поля введення даних, змушуючи базу даних виконувати несанкціоновані операції (видалення, читання конфіденційних даних).

– XSS (Cross-Site Scripting). Впровадження шкідливого клієнтського скрипту в веб-сторінку, який виконується в браузері інших користувачів. Це дозволяє викрадати сесійні файли cookie або інші дані користувачів.

– Небезпечна десеріалізація. Перетворення даних зі збереженого (серіалізованого) формату назад в об'єкти може призвести до віддаленого виконання коду.

– Некоректний контроль доступу. Зловмисник отримує доступ до функцій або даних, до яких не повинен мати доступу (наприклад, перегляд даних інших користувачів або доступ до адміністративних функцій).

– Вразливості XML External Entities (XXE): атаки, які використовують недоліки у парсерах XML.

2. Проблеми автентифікації та сесій. Ці проблеми пов'язані з тим, як додаток ідентифікує користувачів і керує їхніми сеансами.

– Некоректна автентифікація. Використання слабких, передбачуваних паролів, відсутність двофакторної автентифікації (2FA) або неправильне зберігання хешів паролів.

– Управління сесіями. Викрадення або фіксація сесійних ідентифікаторів (Session IDs) через незахищені канали (HTTP замість HTTPS) або неправильне налаштування файлів cookie.

3. Проблеми конфігурації та розгортання. Ці вразливості виникають не в коді, а в навколишньому середовищі або налаштуваннях.

– Неправильна конфігурація безпеки. Використання налаштувань за замовчуванням, увімкнення невикористовуваних портів чи сервісів, неповні права доступу до файлів/каталогів.

– Уразливості в сторонніх компонентах. Використання застарілих або вразливих бібліотек та фреймворків. Оскільки сучасні веб-додатки на 80% складаються з відкритого вихідного коду (Open Source), моніторинг цих залежностей є складним завданням.

– Витік інформації. Додаток може ненавмисно розкривати конфіденційну інформацію (наприклад, версії програмного забезпечення, системні шляхи, повідомлення про помилки).

4. Атаки на доступність. Хоча ці атаки спрямовані на інфраструктуру, вони критично впливають на доступність веб-додатку.

– DdoS. Перевантаження веб-додатку або інфраструктури величезною кількістю трафіку, що робить його недоступним для легітимних користувачів.

– Application-Level DdoS. Атаки, які фокусуються на вразливих, ресурсомістких функціях додатку (наприклад, складний пошук у базі даних), щоб вивести його з ладу мінімальною кількістю запитів.

Світ швидко змінюється, і все більше аспектів нашого життя переходять до цифрової сфери. Від оплати комунальних послуг до покупок продуктів і оформлення документів, все це можна зробити віддалено, не виходячи з дому. Люди передають все більше особистої інформації на сервери компаній та державних організацій, таку як номери паспорта або кредитних карток.

На рисунку 3.1 можна побачити веб-додаток для аналізу кредитних пропозицій.

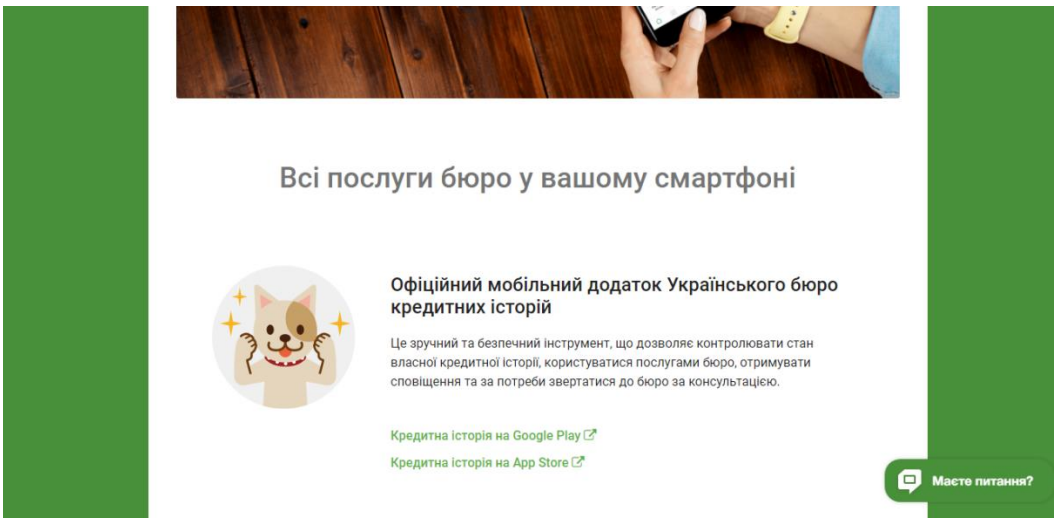

Рисунок 3.1 – Веб-додаток для аналізу кредитних пропозицій фінансового ринку

Витік цих даних може призвести до серйозних наслідків, таких як крадіжка коштів або навіть втрата особистості. Тому безпека даних користувачів та інформаційної системи в цілому є найважливішою метою будь-якої компанії. Втрата даних обов'язково призведе до руйнування іміджу бізнесу та відсутності довіри до нього з боку потенційних клієнтів, тому кожна компанія має ставити перед собою чіткі завдання. Завдання аналізованого веб-додатку наведено на рисунку 3.2.

Рисунок 3.2 – Завдання веб-додатку для аналізу кредитних пропозицій фінансового ринку

Програмне забезпечення з відкритим кодом (Open Source Software, OSS) стало невід'ємною частиною роботи сучасних розробників. За своєю суттю, OSS – це програмне забезпечення, вихідний код якого опубліковано під ліцензією, що надає будь-якій особі право вільно його використовувати, вивчати, змінювати та розповсюджувати для будь-яких цілей.

OSS є яскравим прикладом відкритої співпраці, оскільки його часто створює спільнота незалежних інженерів. Перевага його використання очевидна: більшість сучасних операційних систем, мов програмування, бібліотек і модулів є відкритими. Це значно прискорило розвиток програмної інженерії, оскільки розробникам більше не потрібно створювати функціонал "з нуля", а можна ефективно інтегрувати або модифікувати вже наявні рішення.

Попри очевидні переваги, активне використання програмного забезпечення з відкритим кодом (OSS) несе в собі значні ризики для безпеки інформаційної системи, оскільки сторонні бібліотеки можуть містити фрагменти, що призводять до витоку конфіденційної інформації. Розробники, які використовують OSS, значною мірою покладаються на те, що творці цих проєктів мінімізували безпекові загрози. З огляду на це, додання будь-якої нової бібліотеки в код застосунку вимагає максимально відповідального підходу.

Для ефективного захисту критично важливо впровадити спеціалізовані інструменти сканування, які аналізують як сам додаток, так і його залежності на предмет вразливостей. Ці рішення дозволяють виявити потенційні загрози в коді з відкритим вихідним кодом та вказують на необхідність термінового оновлення до найновіших версій. Оскільки такі інструменти мають доступ до комерційної таємниці (коду ІС та даних), рекомендується обирати лише перевірені часом та відомі рішення, які використовують великі компанії. Це є запорукою надійної безпеки даних та коду інформаційної системи.

3.2 Огляд інструментів для визначення можливих вразливостей

Для визначення можливих вразливостей у програмному забезпеченні та інформаційних системах використовується комплекс інструментів, які поділяються на кілька категорій залежно від методу аналізу та етапу життєвого циклу розробки (SDLC), на якому вони застосовуються. Ці інструменти є основою підходу DevSecOps, який інтегрує безпеку в усі етапи розробки.

1. Статичний аналіз безпеки застосунків (SAST) аналізує вихідний код, бінарні файли або байт-код програми без її фактичного виконання. Це дозволяє виявляти вразливості на ранніх етапах розробки (до розгортання). Наприклад, Checkmarx (рис.3.3), SonarQube, Bandit.

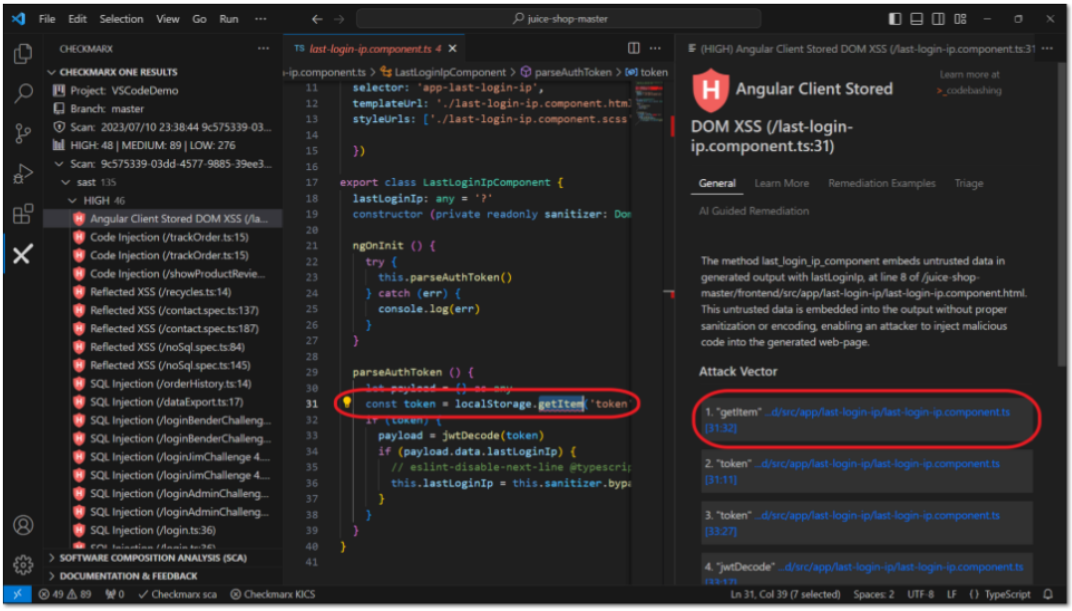

Рисунок 3.3 – Панель результатів Checkmarx

2. Динамічний аналіз безпеки застосунків DAST (Dynamic Application Security Testing) аналізує застосунок під час його виконання (у працюючому стані) шляхом імітації дій зловмисника, які надсилають запити на вхідні та вихідні дані. DAST бачить застосунок «ззовні». Наприклад, OWASP ZAP, Burp Suite, Acunetix / Netsparker.

3. Аналіз складу програмного забезпечення (Software Composition Analysis) сканує застосунок на наявність сторонніх бібліотек, модулів та залежностей з відкритим кодом (OSS) та перевіряє їх на відповідність відомим базам вразливостей. Наприклад, Dependabot, OWASP Dependency-Check, Snyk.

3.3 SQL-ін’єкцій та захист від них

SQL-ін'єкція – це поширена вразливість веб-безпеки, яка дає змогу зловмиснику втручатися у запити, що надсилаються додатком до його бази даних. Як правило, ця атака дозволяє несанкціоновано переглядати дані, які мають бути недоступними (наприклад, інформацію інших користувачів або системні дані). У багатьох критичних випадках зловмисник може також змінювати або видаляти ці дані, що призводить до постійних негативних змін у вмісті чи поведінці програми.

У деяких ситуаціях існує можливість застосувати атаку ін'єкцій SQL, щоб поставити під загрозу базовий сервер чи іншу інфраструктуру. Успішна атака ін'єкцій SQL може призвести до несанкціонованого доступу до конфіденційних даних, таких як паролі, дані кредитної картки або особиста інформація користувача.

В дослідженні було перевірено роботу підриву логіки програми за допомогою ін'єкцій SQL, де є можливість змінити запит, задля перешкоджання логіки програми, але також існує велика різноманітність і інших вразливих ситуацій, атак та методів ін'єкцій SQL, які виникають у різних ситуаціях. Деякі поширені приклади введення SQL включають:

– Отримання прихованих даних, де можна змінити SQL-запит, щоб повернути додаткові результати.

– UNION-атаки, де можливо отримати дані з різних таблиць баз даних.

– Вивчення бази даних, де можливо отримати інформацію про версію та структуру бази даних.

– Сліпа ін'єкція SQL, коли результати запиту, яким керує зловмисник, не відповідають результатам програми.

Підрив логіки програми

Розглянемо програму, яка дозволяє користувачам входити з іменем користувача та паролем. Якщо користувач подає ім'я користувача Andrew та пароль 123456, програма перевіряє облікові дані, виконуючи наступний SQL-запит:

SELECT * FROM accounts WHERE username = 'andrew' AND password = '123456'

Якщо запит повертає реквізити користувача, то реєстрація успішна. В іншому випадку реєстрація не відбудеться.

Якщо, наприклад, в поле паролю Ввести знак “ ' “ може виникнути помилка, яка надасть нам потрібну інформацію про базу даних чи про файли сайту. Наприклад, як зображено на рисунку 3.3 нижче.

Error: Failure is always an option and this situation proves it	
Line	49
Code	0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''''' at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnotic Information	SELECT * FROM accounts WHERE username='andrew' AND password='''

Рисунок 3.3 – Помилка при відмові в реєстрації користувача

Для перевірки можливості використання SQL коду в полі реєстрації можна в полі password ввести 123456' and 1=1#. Після вводу в разі незахищеності системи реєстрація пройде успішно. Вид запиту після вводу пароля:

SELECT * FROM accounts WHERE username = 'andrew' AND password = '123456' and 1=1#'

Щоб перейти на відповідний аккаунт наприклад andrew, але пароль нам не відомий можна використати or замість and після чого запит буде мати вигляд:

SELECT * FROM accounts WHERE username = 'andrew' OR password = '123456' or 1=1#'

Також існує можливість ввійти, як будь-який користувач, без пароля просто за допомогою поля username. Наприклад, введення імені користувача admin'# та порожнього пароля призводить до наступного запиту:

SELECT * FROM accounts WHERE username = 'admin'# AND password = 'aaaaaa'

Цей запит повертає користувача, ім'я користувача якого є, admin і успішно входить в систему. Так як після 'admin' стоїть знак коментаря #, то все, що йде після нього, буде ігноруватись. Тому запит в дійсності буде таким:

SELECT * FROM accounts WHERE username = 'admin'#

Підрив логіки програми з більш складнішим захистом.

Візьмем більш складну систему яка блокує різні символи крім букв. Тоді при вводі будь-якого знаку буде видаватись така помилка (рис.3.4):

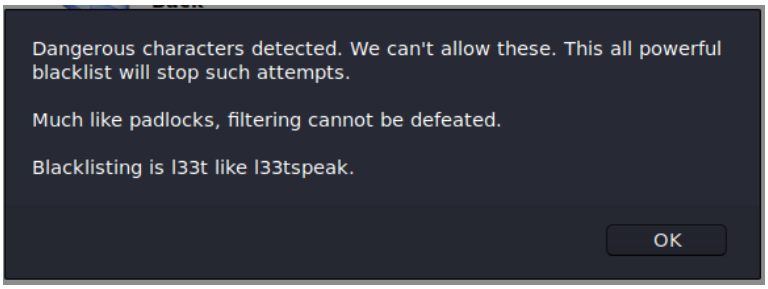

Рисунок 3.4 – Помилка при вводі на стороні клієнта

Дана помилка відбувається тільки на стороні клієнта, і до серверу запит не відправляється, тому даним способом доступ до бази не буде отримано. Але, якщо будуть введені коректні символи, то зв’язок з базою відбудеться хоча б, щоб перевірити параметри авторизації. На даному етапі можна скористатися програмою Burp Suite та переглянути які пакети надсилаються (рис.3.5):

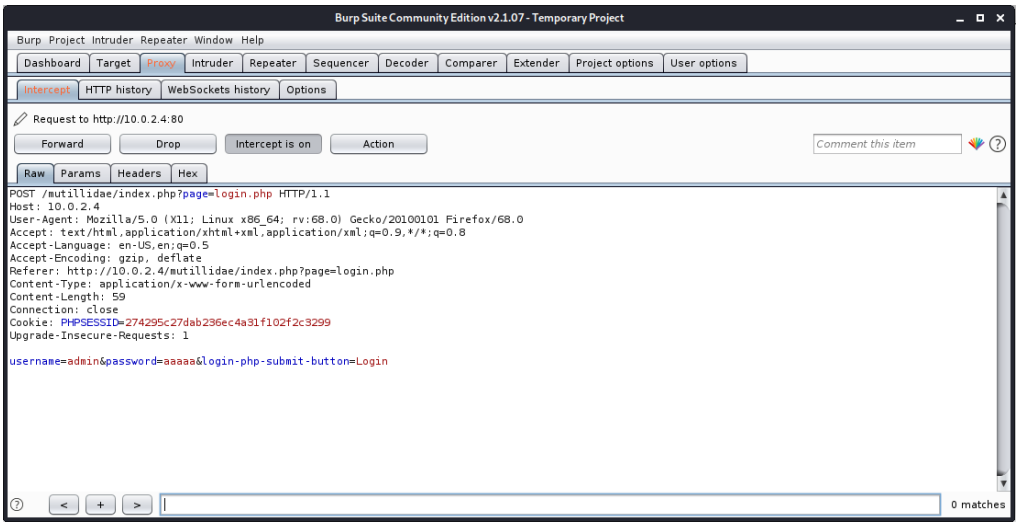

Рисунок 3.5 – Інтерфейс програми Burp Suite

Дана програма може «притримати» пакети, які вже вийшли з клієнта, а попередня помилка не буде завадою для отримання доступу до бази. В цій програмі також можливо дещо підправити відправлені дані, а саме переправити пароль на вже відомий код 123456' or 1=1# (рис. 3.6).

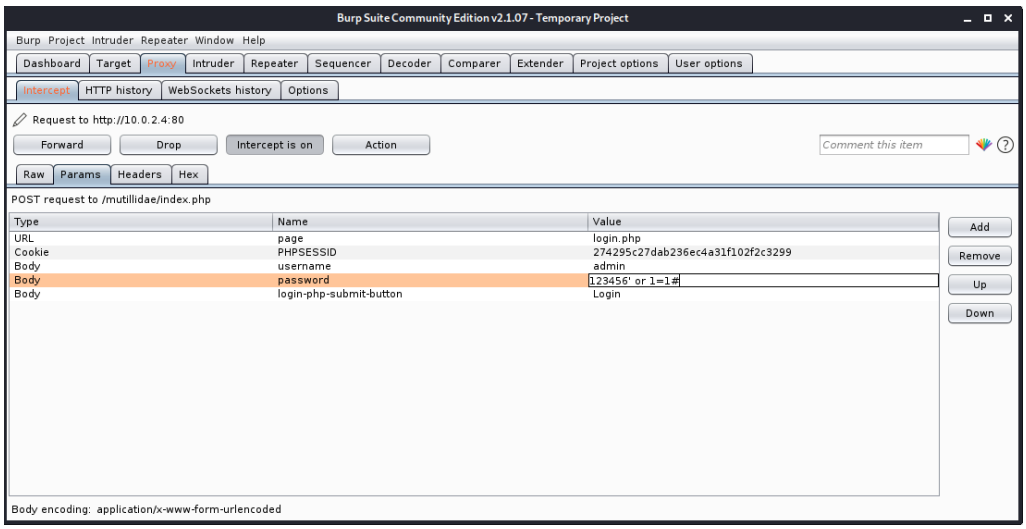

Рисунок 3.6 – Деякі можливості програмного продукту Burp Suite

Більшість уразливостей ін'єкцій SQL можна швидко та надійно знайти за допомогою веб-сканера вразливості Burp Suite .

Ін'єкцію SQL можна виявити вручну за допомогою систематичного набору тестів проти кожної точки входу в програму. Зазвичай це стосується:

– Подання символу єдиної цитати та пошуку помилок чи інших аномалій.

– Подання деякого специфічного для SQL синтаксису, який оцінює базове (вихідне) значення точки введення та інше значення та шукає систематичні відмінності в отриманих відповідях додатків.

– Посилають логічні умови, такі, як OR 1=1 і OR 1=2, і шукаємо відмінності у відповідях додатка чи сайту.

– Подання корисних навантажень, призначених для запуску затримок у часі при виконанні в межах SQL-запиту, та пошуку відмінностей у часі, необхідному для відповіді.

– Подання корисних навантажень OAST, призначених для запуску позадіапазонної мережевої взаємодії при виконанні в рамках SQL-запиту, а також для моніторингу будь-яких результуючих взаємодій.

Введення SQL в різні частини запиту. Більшість уразливостей для введення SQL виникає в WHERE пункті SELECT запиту. Цей тип ін'єкції SQL, як правило, добре зрозумілий досвідченим тестерам. Але вразливості введення SQL можуть в принципі виникати в будь-якому місці в межах запиту та в різних типах запитів. Найбільш поширені інші місця, де виникає ін’єкція SQL, це:

– У UPDATE висловлюваннях, в межах оновлених значень або WHERE пункту.

– У INSERT висловлюваннях у межах вставлених значень.

– У SELECT висловлюваннях, в межах назви таблиці чи стовпця.

– У SELECT заявах, у межах ORDER BY.

Захист сайту від SQL-ін’єкцій. Так як ми вже визначили, що SQL-ін’єкції дуже небезпечні тому потрібно мати надійний захист від даних атак. Тому одним з виходів використовувати PHP Data Objects або PDO. PDO (PHP Data Objects) - рівень абстракції для запитів вашої бази даних і є приголомшливою альтернативою MySQLi, оскільки він підтримує 12 різних драйверів баз даних. MySQL - це найпопулярніша база даних. Вона також дуже добре поєднується з PHP, саме тому ця пара технологій добре підтримується у світі PHP.

Якщо фактично знаєте, що єдиною базою даних SQL, якими ви користуєтесь, є або MySQL, або MariaDB, слід вибрати PDO.

Існує думка, що головна перевага PDO полягає в тому, що він переноситься з бази даних в базу даних але насправді це не є так. Насправді, дуже рідко приймається рішення щодо переключення баз даних на конкретний проект, а саме для компаній середнього рівня дана перевага є недоречною. Незважаючи на це, зазвичай, як правило, вважають за краще дотримуватися поточної технології, яка використовується, якщо тільки немає обґрунтованих причин втрачати значну кількість часу та грошей для переносу.

Справжня перевага PDO полягає в тому, що використовується практично подібний API для будь-якої з безлічі баз даних, які він підтримує, тому вам не потрібно вивчати нову для кожної. Названі параметри також, безсумнівно, є величезною перевагою для PDO, оскільки ви можете повторно використовувати однакові значення в різних місцях запитів. На жаль, ви не можете використовувати одні і ті ж названі параметри не один раз із вимкненим режимом емуляції.

Робота підготовлених заявок PDO. Для простішого пояснення, підготовлені заяви PDO працюють так:

1. Підготуйте SQL-запит із порожніми значеннями, як заповнювачі або знаком питання, або ім'ям змінної з двокрапкою, що передує йому, для кожного значення.
2. Прив’яжіть значення або змінні до заповнювачів
3. Виконувати запит одночасно.

Недоліки PDO. Однак при використанні визначених виразів спільно з PDO необхідно знати деякі нюанси, щоб уникнути неприємних ситуацій. Наприклад, в MySQL клієнта деякі запити, складені за допомогою визначених виразів, тому не можуть бути виконані, а так само вони не використовують кеш, що може уповільнити роботу вашого web-додатка.

Гарантована безпека при використанні визначених виразів звучить добре, але розробники не повинні приймати PDO і інші види абстракції, зумовлені

вираженням, як абсолютний захист від злому. Будь-які вхідні дані повинні перевірятися, PDO - додаткова лінія оборони. Це розширення не закриває все безліч вразливостей, за допомогою яких може бути завдано шкоди вашій інформації, але в той же час, PDO непогано справляється з питанням запобігання SQL ін'єкцій.

Висновок до розділу 3

Створення сучасної програми без використання сторонніх бібліотек практично неможливе. Використання програмного забезпечення з відкритим кодом стало популярним і допомагає розробникам створювати складні програми, які ми бачимо в сучасному Інтернеті.

Однак, використання сторонніх бібліотек також має свої ризики для безпеки. Навіть якщо дотримуватися найкращих практик розробки програмного забезпечення і регулярно перевіряти його безпеку, одна вразливість у бібліотеці може призвести до ризику програми.

Статистика показує, що більшість компаній використовують програмне забезпечення з відкритим кодом. Проте, багато веб-розгортань, що залежать від таких бібліотек, стають вразливими до різних загроз безпеки. Розробники іноді занадто залежать від бібліотек з відкритим кодом для різних аспектів своїх програм. Але важко вручну відстежувати всі потенційні вразливості в інформаційній системі. Навіть якщо перевіряти всі залежності в надійному джерелі, це не гарантує, що нові вразливості не будуть знайдені в майбутньому.

Існують різні інструменти, які допомагають управляти вразливими місцями з відкритим кодом в програмі та її залежностях. Наприклад, OSSIndex від OWASP – це інструмент командного рядка, який допомагає моніторити безпеку PHP, Java, .Net і Ruby. Є також комерційні організації, які надають аналіз безпеки програмного забезпечення та інтегруються з різними службами зберігання коду. Хоча неможливо гарантувати абсолютну безпеку, постійний

аналіз, оновлення бібліотек і виправлення вразливостей допомагають зменшити ризики втрати даних. Втім, варто пам’ятати, що повна безпека недосяжна [7].

ВИСНОВКИ

Проаналізований веб-застосунок для аналізу кредитних пропозицій фінансового ринку та його залежності на рахунок можливих вразливостей безпеки, були розглянуті найпоширеніші рішення і вибраний інструмент сканування та усунення вразливостей Burp Suite. Інтеграція цього інструменту в додаток дозволила виявити загрози безпеці даних компанії та користувачів і мінімізувати можливість їх виникнення.

Була здійснена мінімізація ризиків втрати даних користувачів та компанії. Інструмент моніторингу продуктивності додатку Prometheus та Grafana інтегрований у кваліфікаційну роботу, з його використанням скорочується час виконання фрагментів програмного коду. Інтегрований інструмент для відстеження вразливостей у безпеці додатків і його залежностей за допомогою нього знижує ризик втрати даних користувача та компанії.

Були проаналізовані швидкості виконання програмного коду додатку завдяки рішенням моніторингу продуктивності додатків (APM) компанії можуть відстежувати, чи відповідає їхня інформаційна система стандартам продуктивності, виявляти помилки та потенційні проблеми та забезпечувати бездоганну роботу користувачів шляхом ретельного моніторингу своєї інфраструктури. Найкращі рішення APM надають командам розробників інформацію, необхідну для прискорення програми та підвищення пропускної здатності, а також для виявлення й усунення проблем із продуктивністю, перш ніж вони вплинуть на кінцевого користувача.

Був проведений рефакторинг та зменшення часу виконання підпрограм інформаційної системи. Проаналізовано найпопулярніші на сьогодні засоби моніторингу інформаційної системи та обрано як рішення для подальшого використання. З його допомогою були оптимізовані повільні запити до бази даних, рефакторинг фрагментів програмного коду, що в свою чергу сприяло зменшенню часу виконання підпрограм.

Щоб забезпечити швидкість і стабільність інформаційної системи, моніторинг і рефакторинг повинні бути частиною щоденної роботи команди інженерів.

Так, потрібен час, який команда могла б витратити на розробку нового функціоналу, але в довгостроковій перспективі це принесе більше користі для бізнесу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Welcome to - Digital Library NAES of Ukraine. URL: https://lib.iitta.gov.ua/725877/1/Пророк_стаття1_2021%20(1).pdf (дата звернення: 27.11.2022).

2. Topic: Mortgage industry in the U.S. Statista. URL: https://www.statista.com/topics/1685/mortgage-industry-ofthe-united-states (date of access: 11.12.2022).

3. Is higher education still a wise investment? Evidence on rising student loan debt in the U.S. Psychosociological Issues in Human Resource Management. 2018. Vol. 5, no. 1. P. 235. URL: https://doi.org/10.22381/pihrm5120179 (date of access: 10.01.2023).

4. Install new relic ruby agent. Ukraine. URL: https://docs.newrelic.com/ docs/agents/ruby-agent/installation/install-newrelic-ruby-agent (date of access: 07.02.2023).

5. Ruby agent configuration. Ukraine. URL: https://docs.newrelic.com/ docs/agents/ruby-agent/configuration/rubyagent-configuration (date of access: 09.03.2023).

6. Its an open source world. Ukraine. URL: https://www.zdnet.com/article/its-an-open-source-world-78-percent-ofcompanies-run-open-source-software/ (date of access: 22.04.2023).

7. Монітор серверів і застосунків – Спостереження , власний хостинг | SolarWinds. Спостереження, управління базами даних та IT-послугами | SolarWinds . URL: https://www.solarwinds.com/server-application-monitor (дата доступу: 27.05.2023).

8. Платформа Splunk. URL: https://surl.lt/bufagw (дата доступу: 09.03.2023).

9. Рейтинг популярних сайтів за січень 2020. Ukraine. URL: https://tns-ua.com/news/reyting-populyarnih-saytiv-za-sichen2020. (дата звернення: 27.05.2023).