DOI 10.31891/2307-5732-2024-341-5-48
УДК 004.85:004.056

КРАВЧУК НАЗАР
Національний університет «Львівська політехніка»
https://orcid.org/0009-0002-1008-4765
e-mail: nazar.kravchuk.official@gmail.com
КОРОБЕЙНІКОВА ТЕТЯНА
Національний університет «Львівська політехніка»
https://orcid.org/0000-0003-2487-8742
e-mail: tetianakorobeinikova@gmail.com

БЕЗПЕЧНИЙ ДОСТУП ДО СЕРВЕРІВ ІНФОРМАЦІЙНИХ СИСТЕМ, ЗАБЕЗПЕЧЕНИЙ ML-МОДЕЛЛЮ ДЛЯ БЛОКУВАННЯ ШКІДЛИВИХ ЗАПИТІВ

В роботі наведено детальне дослідження використання алгоритму k-найближчих сусідів (KNN) для класифікації та ідентифікації різних типів кібератак, зокрема SQL-ін'єкцій (SQLi), у системах SCADA. SQL-ін'єкції передбачають впровадження зловмисного коду SQL у форми для маніпулювання запитами до бази даних, потенційно в обхід автентифікації або отримання неавторизованих даних. У роботі пояснюється, як можна використати уразливості SQL-ін'єкцій через недостатньо відфільтровані поля введення, що може призвести до потенційних порушень даних.

Ключові слова: KNN, SQL-ін'єкція, системи SCADA, захист даних, кібератаки, класифікація.

KRAVCHUK NAZAR, KOROBEINIKOVA TETIANA
Lviv National Technical University

SECURE ACCESS TO INFORMATION SYSTEM SERVERS, ENABLED BY AN ML MODEL FOR BLOCKING MALICIOUS REQUESTS

The article presents a detailed exploration of using the k-Nearest Neighbors (KNN) algorithm to classify and identify various types of cyberattacks, particularly SQL Injection (SQLi) attacks, within SCADA (Supervisory Control and Data Acquisition) systems.

This work addresses the need to enhance server infrastructure security in SCADA systems by mitigating the risks posed by harmful requests, such as SQL injections. SCADA systems are crucial in managing industrial processes, making their servers prime cyberattack targets. Attackers often exploit vulnerabilities in server applications by injecting malicious requests through input fields or URLs, potentially gaining access to sensitive data or disrupting system operations. To address this issue, the study proposes a machine learning-based approach using the k-nearest neighbors (KNN) algorithm to detect and block harmful SQL requests. The KNN algorithm is employed to classify and identify different types of cyberattacks by comparing new attack attempts with previously observed attack patterns. By analyzing specific attributes related to each attack, the KNN method evaluates the level of threat based on proximity metrics. The proposed approach helps classify SQL injection attempts, which involve manipulating SQL code to bypass authentication or extract unauthorized data. The study demonstrates how KNN can effectively distinguish harmful SQL requests from benign ones by calculating the Euclidean distance between the new attack and historical cases.

Furthermore, the article emphasizes the importance of implementing rapid and accurate detection methods for protecting server infrastructure in industrial environments. The KNN algorithm, in this context, offers a flexible and efficient solution as it adapts to various attack scenarios, improving the overall resilience of SCADA systems to cyber threats. The study's findings contribute to the ongoing efforts in cybersecurity, focusing on integrating machine learning models to strengthen the protection of critical assets in industrial control systems.

This work aims to develop protection tools for server-based industrial control systems used in SCADA systems against dangerous requests based on SQL injections, using an ML-trained model for blocking harmful requests through the k-nearest neighbors method.

Keywords: KNN, SQL Injection, SCADA systems, data protection, cyberattacks, classification.

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Системи SCADA зазвичай складаються з двох основних апаратних компонентів: серверів і клієнтських інтерфейсів. Сервери в системі SCADA мають вирішальне значення для збирання даних і контролю процесів. Ці сервери отримують дані з різних джерел, також від з мікроконтролерів та інтелектуальних датчиків. Мікроконтролери часто використовуються для збирання даних безпосередньо з процесу, моніторингу та керування операціями. Інтелектуальні датчики можуть бути підключені безпосередньо до сервера або через проміжні пристрої, такі як станції зв'язку або головні пристрої, які збирають дані з кількох датчиків. Програмовані логічні контролери (ПЛК) зазвичай використовуються в промислових умовах для збирання даних і керування процесами, при цьому сервери взаємодіють із цими ПЛК для керування даними і для оброблення самих даних. Клієнт, що має доступ до мережі, отримує та відображає дані з сервера, таким чином забезпечує взаємодію з людиною-оператором. Відомо, що сервери відіграють центральну роль у обробленні та зберіганні даних і можуть бути основними цілями для кібератак. Зловмисники можуть використовувати уразливості в серверних програмах, щоб вводити шкідливі запити через поля введення або URL-адреси, потенційно отримувати доступ до баз даних, які зберігають чутливу інформацію. Такі вразливості можуть поставити під загрозу цілісність і безпеку всієї системи SCADA, що робить важливим впровадження надійних заходів безпеки для захисту від впровадження SQL та інших кіберзагроз.

З цієї причини, поточне дослідження зосереджуватиметься на інтеграції засобів захисту серверних промислових систем управління (Industrial Control Systems – ICS), які використовуються в системах SCADA (Supervisory Control and Data Acquisition) для управління промисловими процесами шляхом моніторингу і контролю виробничих одиниць, від шкідливих запитів по типу тих, що були описані вище, на основі методу машинного навчання k-найближчих сусідів (k-nearest neighbours – KNN).

Аналіз досліджень та публікацій

У науково-дослідницькому просторі сьогодення з'являються роботи, присвячені винаходу та аналізу методології по розробці захисної інфраструктури для серверів інформаційних систем від шкідливих запитів. Особлива увага приділяється використанню штучного інтелекту та машинного навчання для виявлення та блокування таких загроз, як SQL-ін'єкції, атаки на відмову в обслуговуванні (DDoS), фішинг і спроби несанкціонованого доступу. Такі підходи дозволяють створювати динамічні, здатні до самонавчання системи захисту, які можуть адаптуватися до нових типів загроз, і таким чином, покращити загальну безпеку та стійкість інформаційних систем до кібератак.

У роботі [1] досліджуються ризики безпеки, пов'язані з атаками ін'єкцій P2SQL (Prompt-to-SQL – SQL ін'єкція з підсказками), і представлений набір засобів захисту. Ці атаки можуть бути особливо небезпечними у веб-додатках, інтегрованих у LLM (Large Language Model – велика мовна модель), що призводить до знищення даних і порушення конфіденційності. При використанні Langchain як платформу для розробки чат-ботів, аналізуються різні типи атак P2SQL і демонструється, що в цих сценаріях можна використовувати найсучасніші моделі LLM. Дослідження передбачає оцінку сприйнятливості кількох LLM до таких атак, це дає змогу виявити значні вразливості в системах на основі Langchain. У відповідь на ці вразливості пропонуються кілька стратегій захисту великих мовних моделей, які враховують покращення привілеїв баз даних SQL–серверів, переписування запитів SQL, використання перевірки на основі LLM і забезпечення швидкого попереднього завантаження даних. Ці методи впроваджуються та перевіряються за допомогою експериментів із застосуванням у реальному світі. Робота відкриває нові шляхи для майбутніх досліджень, зосереджених на виявленні нових вразливостей P2SQL, пропонуванні нових засобів захисту, зменшенні накладних витрат на ці засоби захисту, автоматизації дослідження вразливостей P2SQL і розробці зручної та модульної структури для захисту від атак P2SQL.Дослідження сприяє виявленню та аналізу вразливостей впровадження P2SQL, і таким чином пропонує практичні захисні заходи та підтверджує ці заходи шляхом емпіричного тестування, забезпечує основу для підвищення безпеки веб-додатків, інтегрованих у LLM, від загроз впровадження SQL.

Дослідження роботи [2] зосереджувалося на розробці агентів навчання з підкріпленням (reinforcement learning – RL) для ефективного усунення вразливостей впровадження SQL–ін'єкцій (SQL Injection – SQLi). Агенти були навчені в синтетичному середовищі, спеціально розробленому для імітації різних сценаріїв SQLi, які містили сліпі ін'єкції на основі стека, об'єднання, логічного значення, сліпі ін'єкції на основі помилок і часу. Ключовим внеском є розробка підходу до передавального навчання (Transfer learning), який дозволяє агентам RL узагальнювати свої знання від одного типу вразливості SQLi до іншого. Цей підхід покращує адаптивність агентів, дозволяє їм ефективно виконувати різні сценарії атак SQLi, не вимагає тривалого перенавчання для кожного конкретного типу. Крім того, було представлено новий метод інтерпретації повідомлень сервера, який зменшує залежність від експертів домену. Цей метод спрощує процес розуміння та реагування на відгуки сервера під час атак, роблячи агентів більш автономними та ефективними. Щоб оцінити ефективність агентів RL, було проведене широке моделювання в синтетичному середовищі. Ці симуляції оцінювали продуктивність агентів у виявленні та пом'якшенні різних вразливостей SQLi. Результати показали, що агенти були ефективними в обробленні ряду сценаріїв SQLi. Проте спостерігалися деякі обмеження, зокрема у боротьбі з атаками, які представляли вищий рівень невизначеності або складності.

Крім того, варто зазначити праці українських науковців: Максима Кавецького, Олександра Сєвєрінова, Романа Гвоздьова, Антона Смірнова [3] щодо необхідності виявлення та протидії атакам типу DOS/DDOS, що є серйозною загрозою для сучасних інформаційних систем; Людмилу Асєєву [4] щодо управління інформаційною безпекою підприємства з використанням методів машинного навчання та нечіткої логіки; Євгена Іваніщенка, Мілану Сабліну, Катерину Кравчук (та ін.) [5] щодо інтеграції технологій машинного навчання в системи кібербезпеки; Романа Банаха, Андріана Піскозуба, Івана Опірського (та ін.) [6] в галузі моніторингу, аналізу пакетів мережі ethernet і зберігання даних на основі часових рядів. Серед іноземних науковців можна згадати таких: Хачам С., Учан О. [7], Нікхіта М. Джаббар М.А. [8], Шах С., Крішна В., Картік В., Гурурадж Р. [9], Чжан Б., Рен Р., Лю Д., Цзян М., Жень Дж., Лі Дж. [10] , Кукутла Т. [11], Шарма Ст, Кумар С. [12], Умар До., Бакар А., Зулзаліл Х., Адмодісастро Н., Абдулла М. [13], Кулсум Ф., Нареджо С., Мехмуд З., Чоудрі Х., Батт А., Башир А. [14], Сільвестре А., Медейрос І., Мордідо А. [15], Креспо-Мартінес І., Вега А., Герреро-Ігерас А., Рієго Ст, Альварес-Апарісіо К., Фернандес К. [16], Касим О. [17], Падма Дж., Равішанкар Н., Раджу М.Б., Раві Н. [18], Хадабі А., Ельсамані Е., Абдаллах А., Ельхабоб Р. [19] та інших.

Проте, беручи до уваги роботи вище зазначених авторі, питання, пов'язане з методологією з розробки засобів захисту серверних систем управління на основі методів машинного навчання, все ще залишається недостатньо дослідженим та потребує подальшого опрацювання.

Формулювання мети роботи

Метою роботи є розробка засобів захисту серверних промислових систем управління, які використовуються в системах SCADA, від небезпечних запитів на базі SQL-ін'єкцій, з використанням ML-тренованої моделі блокування небезпечних запитів з використанням методу k-найближчих сусідів.

Виклад основного матеріалу

Алгоритм K-nearest neighbors (KNN) – простий і ефективний метод машинного навчання, який використовується для вирішення задач класифікації та регресії. Новий об'єкт класифікується на основі його близькості до вже відомих об'єктів із тренувальної вибірки. KNN є інстанційним алгоритмом, оскільки не будує

явної моделі під час тренування, а зберігає всю тренувальну вибірку. Класифікація чи регресія відбуваються безпосередньо під час оброблення нових даних. Загальний принцип роботи алгоритму KNN можна описати завдяки прикладу, згідно якому, червоний п'ятикутник має класифікуватися як синій квадрат або як зелений трикутник [14] (рис. 1). Якщо k = 3, то він класифікується як зелений трикутник, тому що всередині меншого кола 2 трикутника і тільки 1 квадрат. Якщо k = 5, то він класифікуватиметься як червоний квадрат (3 квадрата проти 2-ох трикутників всередині більшого кола).

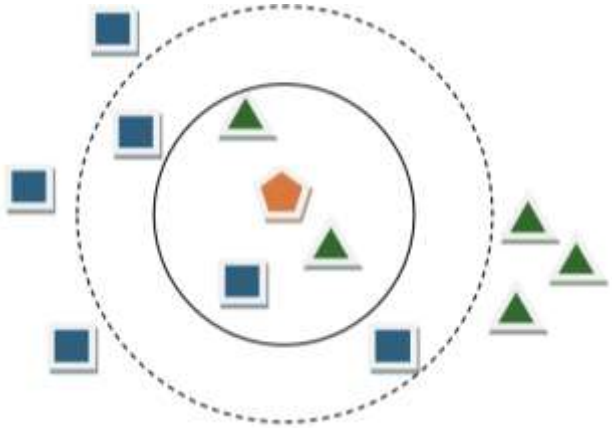

Рис. 1. Схематичне зображення методу k-найближчих сусідів

Небезпечні запити до серверної інфраструктури зазвичай передбачають спроби використати вразливі місця, отримати несанкціонований доступ або порушити нормальні операції, типові різновиди яких містять:

- SQLi, при якій зловмисником додається SQL-код до введення форми, із маніпулюванням запитом, щоб обійти автентифікацію або отримати неавторизовані дані. SQL ін'єкція може статися через вхідні дані, які не були належним чином відфільтровані. Якщо на веб-ресурсі немає належної фільтрації URL-адреси, він стає сприйнятливим до атак SQL-ін'єкцій. Наприклад, якщо веб-додаток дозволяє користувачеві вводити дані безпосередньо в запит без відповідної перевірки, зловмисник може використати це, впровадивши код SQL через URL-адресу. Це може передбачати додавання параметрів, які змінюють призначений запит до бази даних, потенційно повертає або змінює небажані дані. Скажімо, якщо URL-адреса структурована так, щоб передавати ідентифікатор для отримання певних даних, зловмисник може маніпулювати вхідними даними, щоб виконати запит, який відкриває цілі таблиці бази даних або навіть змінює дані;
- міжсайтовий скриптинг (Cross-Site Scripting – XSS), в ході якого шкідливі сценарії впроваджуються на веб-сторінки, які переглядають інші користувачі, що дозволяє зловмиснику викрасти файли cookie, маркери сеансу або іншу конфіденційну інформацію;
- віддалене додавання файлів (Remote File Inclusion – RFI), як правило, сценарію, який потім виконується на сервері і дозволяє контролювати його або викрадати дані;
- додавання локальних файлів (Local File Inclusion – LFI), принцип якого полягає в тому, що зловмисник отримує доступ до локальних файлів на сервері, таких як конфігураційні файли або файли паролів, що призводить до крадіжки даних або подальшого використання;
- Path Traversal (Обхід шляху), що відповідає переміщенню каталогами на сервері, з отриманням доступу до файлів поза передбаченою структурою каталогів, та потенційним відкриванням конфіденційних даних;
- міжсайтова підробка запитів (Cross-Site Request Forgery – CSRF), основною метою якої є змушення користувача обманом, який увійшов у систему, виконувати дії без його згоди, наприклад переказати кошти чи змінити дані облікового запису.

Алгоритм KNN може використовуватися для класифікації та ідентифікації різних типів кібератак у серверах систем SCADA шляхом порівняння нових спроб атак з раніше спостережуваними моделями. Ця методологія передбачає аналіз конкретних атрибутів, пов'язаних з кожним типом атаки, і застосування принципів KNN для оцінки рівнів загрози на основі показників близькості. У контексті атак SQL ін'єкції, які передбачають введення зловмисних запитів SQL для маніпулювання або вилучення даних з бази даних, KNN може класифікувати ці атаки, шляхом оцінювання атрибутів, наприклад, повідомлення про помилки SQL, наявність ключових слів SQL і відповідь сервера. Близькість між новою спробою впровадження SQL та історичними випадками обчислюється за допомогою формули евклідової відстані (1):

$$d(T,S) = \sqrt{\sum_{i=1}^{n} w_i * (x_i - s_i)^2}, \quad (1)$$

де T – нова атака; S – історична атака; w_i – ваги атрибутів; x_i і s_i – відповідають значенням атрибутів для T і S відповідно.

Після обчислення відстаней алгоритм KNN класифікує нову атаку на основі мітки більшості серед її найближчих k-сусідів, що дозволяє ефективно виявляти та запобігати загрозам SQL-ін'єкції. Таким чином, під час застосування KNN для виявлення SQL-ін'єкцій у серверах систем SCADA процес передбачає класифікацію нових SQL-запитів на основі їх подібності до відомих шаблонів атак.

Наприклад, якщо значення k (кількість розглянутих найближчих сусідів) встановлено рівним 3, алгоритм може класифікувати новий SQL-запит як SQL-ін'єкцію, якщо він дуже схожий на більшість із найближчих 3 відомих шаблонів атак. Якщо серед цих 3 сусідів є 2 відомі шаблони SQL-ін'єкції, новий запит класифікується як атака SQL-ін'єкції. І навпаки, якщо k встановлено на 5 і є 3 відомі атаки проти 2 доброякісних запитів у найближчих сусідів, новий запит класифікується як атака SQL-ін'єкції на основі більшості голосів.

Однак, традиційний підхід KNN містить лінійний процес пошуку, що призводить до тимчасової складності $O(nd)$, де nвідповідає кількості примірників і d–розмірності набору даних. Алгоритм поточного дослідження, на противагу, сортує підмножину даних на основі вибраних параметрів, створює умови для швидшого пошуку за допомогою двійкового пошуку, що має вирішальне значення для своєчасного виявлення спроб SQL–ін'єкцій. Алгоритм починається з вибору підмножини параметрів із журналів даних сервера SCADA, зосереджуючись на тих, які найбільше стосуються виявлення аномалій, що вказують на SQL–ін'єкції. Найважливіший вимір визначається за допомогою взаємної інформації, яка вимірює залежність між функціями та ймовірність SQL–ін'єкції.

Навчання моделі KNN для виявлення SQL-ін'єкцій на серверах SCADA систем починається зі збирання різноманітних даних запитів SQL. Цей набір даних містить зразки атак і легітимні запити. Дані про атаки отримуються з історичних журналів, відомих баз даних атаки та змодельованих сценаріїв. Після збирання набору даних, відповідні функції витягуються із запитів SQL. Це передбачає розбиття кожного запиту на маркери, такі як ключові слова SQL, оператори та значення. Основні функції містять наявність певних ключових слів SQL (SELECT, UNION, DROP), структуру запиту та будь-які виявлені шаблони помилок. Ці функції перетворюються в числові формати, придатні для KNN, із використанням кодування one-hot (one-hot encoding) для категоріальних даних і масштабування для числових значень. Цей метод призначає унікальний вектор нулів і одиниць кожній категорії, де довжина вектора відповідає кількості можливих категорій. Кожна категорія представлена вектором з єдиною «1» у позиції, що відповідає цій категорії, тоді як усі інші позиції мають «0». Це кодування гарантує, що кожна категорія є чітко диференційованою, а сума векторних значень завжди дорівнює одиниці, що робить її інтерпретованою як розподіл ймовірностей за категоріями. Якщо нові категорії вводяться пізніше, існуючі вектори оновлюються шляхом додавання додаткового виміру з нульовими значеннями. Схематичне зображення цього типу кодування демонструється на рисунку 2.

SQL-запити з різними атрибутами, такі як ключові слова, оператори та шаблони, кодування one-hot перетворює в двійкові вектори, які KNN може використовувати для вимірювання подібності. Цей процес гарантує, що алгоритм KNN може ефективно порівнювати та класифікувати запити SQL на основі їх закодованих особливостей.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Рис. 2. Схематичне зображення кодування one-hot

Часова складність цього процесу становить $O(nd\log(nd))$ на ітерацію, що призводить до повної складності $O(\mathrm{m}nd\log(nd))$, де m позначає кількість ітерацій. На етапі тестування бінарний пошук зменшує складність пошуку до $O(\log(n))$, з подальшим застосуванням KNN до скороченого набору зі складністю $O(K^2)$. Це призводить до загальної складності тестування $O(m(log(n) + K^2))$.

На додаток до кодування, визначення відповідної функції вартості є важливим для оцінки ефективності моделі KNN. Хоча KNN явно не використовує функцію вартості так само, як нейронні мережі, він покладається на показники відстані для класифікації запитів. Наприклад, евклідова відстань між закодованими ознаками запиту та навчальними вибірками обчислюється для визначення найближчих сусідів. На ефективність моделі впливає те, наскільки добре ці відстані відображають фактичну подібність між запитами.

Оптимізація моделі KNN передбачає вибір оптимальної кількості k найближчих сусідів. Цей вибір безпосередньо впливає на продуктивність моделі у виявленні SQL-ін'єкцій.

Менше значення k може призвести до переобладнання, коли модель є надто чутливою до навчальних даних, у той час як більше значення k може призвести до недостатнього підбору, коли модель узагальнюється занадто широко. Таким чином, перехресна перевірка використовується для визначення найкращого значення k шляхом оцінки точності моделі на різних підмножинах даних.

Під час фази виведення для будь-якого заданого вхідного сигналу, алгоритм використовує бінарний пошук, щоб визначити його позицію в межах відсортованої підмножини. Максимум $K + g$ екземплярів вибрається з обох сторін цієї позиції, де K позначає кількість найближчих сусідів і g – параметр забезпечення

повного охоплення потенційно релевантних випадків. Алгоритм KNN потім застосовується до цього скороченого набору, і основний клас із кількох ітерацій вибирається як остаточний прогноз, який визначає, чи запит несе загрозу SQL-ін'єкції.

Взаємна інформація (mutual information) використовується для визначення параметра, який найбільше корелює з наявністю вразливості до загрози SQL-ін'єкцій. Вона допомагає точно визначити функції в журналах SCADA, які найбільше вказують на аномальну поведінку, керує алгоритмом і зосереджена на найбільш критичних параметрах для сортування та пошуку.

Граційний параметр (grace parameter) G вводиться для врахування специфічних проблем систем SCADA, де потоки даних можуть демонструвати невеликі діапазони варіацій. Трохи розширивши область пошуку, G гарантує, що алгоритм KNN враховує всі потенційно релевантні екземпляри. Виявлення SQL-ін'єкцій полягає у створенні варіантів запитів SQL, які імітують різні типи атак або легітимні запити. Цей підхід допомагає підвищити стійкість моделі шляхом надання їй широкого спектру шаблонів запитів.

Експеримент проводився у середовищі мови програмування Python із використанням таких бібліотек, як NumPy, Pandas і scikit-learn. Для оцінки ефективності K-Nearest Neighbors (KNN) у виявленні атак SQL-ін'єкцій у системах SCADA були використані набори даних, зображені у таблиці 1.

Таблиця 1

Набори даних для оцінки ефективності методу KNNу виявленні атак SQL-ін'єкцій у системах SCADA

Набір даних	Кількість	Опис
SQL_Queries	10,000	Містить SQL-запити, позначені для ін'єкційних і неін'єкційних типів
SCADA_Logs	50,000	Журнали із систем SCADA, які містять SQL-запити, позначені для атаки та нормальні запити
WebApp_Requests	30,000	Журнали запитів HTTP від веб-додатків із спробами SQL-ін'єкцій
Anomaly_Detection	15,000	Дані з різними шаблонами атак, які містять SQL ін'єкції в системах SCADA

** власна розробка автора на основі робіт [15-19]*

Продуктивність стандартного та модифікованого KNN методів порівнювалася з іншими алгоритмами класифікації: Random Forest (RF), AdaBoost. Гіперпараметри для кожного методу: KNN: кількість найближчих сусідів (K) = 3, кількість ітерацій (m) = 3, грація (g) = 0. Випадковий ліс (RF): 100 дерев рішень. AdaBoost: 100 дерев рішень, скорочених на першому рівні. Таблиця 2 показує порівняння згаданих методів класифікації.

Таблиця 2

Порівняння стандартного та модифікованого методів KNN з методами Random Forest (RF) та AdaBoost

Метод	Точність	Відклик	Час тренування	Час тестування
KNN	0.80	0.73	$O(mnd\ log(nd))$	$O(m\ (log(n)\ +\ K^2))$
Random Forest (RF)	0.88	0.77	$O(nd\ log(nd))$	$O(log(n))$
AdaBoost	0.76	0.69	$O(nd)$	$O(nd)$
Модифікований KNN	0.87	0.79	$O(mnd\ log(nd))$	$O(m\ (log(n)\ +\ K^2))$

** власна розробка автора на основі робіт [11-15]*

Експеримент продемонстрував, що KNN є життєздатним методом виявлення SQL-ін'єкцій, і витримує конкурентоспроможність порівняно з іншими алгоритмами класифікації. У середньому KNN досягла точності 80%, а модифікована модель показала невелике покращення. Random Forest забезпечив найвищу точність, але потребував більше обчислень. AdaBoost, хоч і швидший, не працює так добре, як стандартний, або модифікований KNN.

Висновки з даного дослідження і перспективи подальшого розвитку у даному напрямі

Дана стаття присвячена розробці та впровадженню вдосконалених засобів захисту серверних систем, які використовуються в системах SCADA, від небезпечних запитів на базі SQL-ін'єкцій та із використанням ML-тренованої моделі блокування небезпечних запитів з використанням методу k-найближчих сусідів.

У підсумку, використання ML-тренованої моделі виявлення небезпечних запитів у контексті інформаційних серверних систем, а саме на базі алгоритму KNN виявляється цінним інструментом для детекції та класифікації різних типів небезпечних запитів, таких як насамперед ін'єкції SQL у серверних системах SCADA. Із застосуванням принципів KNN, нові спроби атак можна порівняти з інформацією про минулі запити, щоб оцінити рівень загрози.

Цей процес передбачає кодування SQL-запитів у числові формати, такі як двійкові вектори, за допомогою одноразового кодування та обчислення відстаней для виявлення подібності. Ефективність алгоритму залежить від вибору оптимальної кількості найближчих сусідів і відповідного визначення метрики відстані.

Завдяки постійному вдосконаленню та розширенню набору даних KNN може підвищити свою точність у розпізнаванні та пом'якшенні потенційних загроз, тим самим покращує загальну безпеку системи.

Література

1. Pedro R., Castro D., Carreira P., Santos N. (2023) From Prompt Injections to SQL Injection Attacks: How Protected is Your LLM-Integrated Web Application? *arXiv*, available at https://doi.org/10.48550/arXiv.2308.01990.

2. Sommervoll Å., Erdődi L., Zennaro F. (2023) Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents. *International Journal of Information Security*, vol. 23, available at https://doi.org/10.1007/s10207-023-00738-3.

3. Кавецький, М., Сєвєрінов, О., Гвоздьов, Р., & Смірнов, А. (2024) Використання машинного навчання для класифікації атак типу DOS/DDOS. *Радіотехніка*, *2*(217), 55–63. available at: https://doi.org/10.30837/rt.2024.2.217.04.

4. Асєєва, Л. А. (2023) Management of enterprise information security using machine learning and fuzzy logic, doctoral dissertation, Cybersecurity, State University of Information and Communication Technologies, Kyiv, Ukraine.

5. Іваніщенко Євген, Сабліна Мілана, & Кравчук Катерина. (2021) Використання машинного навчання в кібербезпеці. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, *4*(12), 132–142, available at: https://doi.org/10.28925/2663-4023.2021.12.132142.

6. Banakh, R., Piskozub, A., & Opirskyy, I. (2023). Devising a method for detecting “evil twin” attacks on ieee 802.11 networks (wi-fi) with knn classification model. *Eastern-European Journal of Enterprise Technologies*, *123*(9), available at: https://doi.org/10.15587/1729-4061.2023.282131.

7. Hacham S., UÇan O. (2023) Detection of Malicious SQL Injections Using SVM and KNN Algorithms. *7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, Istanbul, Turkiye, pp. 1-5. DOI:10.1109/ISAS60782.2023.10391560.

8. Nikhitha M., JabbarM.A. (2019) K Nearest Neighbor Based Model for Intrusion Detection System. *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8(2), DOI:10.35940/ijrte.B2458.078219.

9. Shah S., Krishna V., Kartheek V., Gururaj R. (2023) A Clustering-Based Approach for Effective Prevention of SQL Injections. *Springer Nature Singapore*, DOI:10.1007/978-981-99-3569-7_29.

10. Zhang B., Ren R., Liu J., Jiang M., Ren J., Li J. (2024) SQLPsdem: A Proxy-based Mechanism towards Detecting, Locating and Preventing Second-Order SQL Injections. *IEEE Transactions on Software Engineering*, vol. 50, no. 7, pp. 1807-1826,. DOI:10.1109/TSE.2024.3400404.

11. Kukutla T. (2023) A Comprehensive Guide to SQL Injection Prevention. *International Center for AI and Cyber Security Research and Innovations (CCRI)*, available at: https://insights2techinfo.com/wp-content/uploads/2023/11/A-Comprehensive-Guide-to-SQL-Injection-Prevention-1-6.pdf.

12. Sharma, V., Kumar, S. (2023) Multi-input MLP and LSTM-Based Neural Network Model for SQL Injection Detection. *Computer Vision and Robotics*. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-19-7892-0_35.

13. Umar K., Bakar A., Zulzalil H., Admodisastro N., Abdullah M. (2016) SQL Injection Attack Roadmap and Fusion. *Indian Journal of Science and Technology*, vol. 9, DOI: 10.17485/ijst/2016/v9i28/97810.

14. Kulsoom F., Narejo S., Mehmood Z., Chaudhry H., Butt A., Bashir A. (2022) A Review of Machine Learning-based Human Activity Recognition for Diverse Applications. *Neural Computing and Applications*. № 34, DOI:10.1007/s00521-022-07665-9.

15. Silvestre A., Medeiros I., Mordido A. (2024) Towards a SQL Injection Vulnerability Detector Based on Session Types. *LASIGE, Departamento de Informatica, Faculdade de Ci ´ encias*, Universidade de Lisboa, Portugal, pp. 711-718. DOI:10.5220/0012732500003687.

16. Crespo-Martínez I., Vega A., Guerrero-Higueras Á., Riego V., Álvarez-Aparicio C., Fernández C. (2022) SQL Injection Attack Detection in Network Flow Data. *Computers & Security*, vol. 127. 103093, DOI:10.1016/j.cose.2023.103093.

17. Kasım Ö. (2021) An ensemble classification-based approach to detect attack level of SQL injections. *Journal of Information Security and Applications*, № 59. 102852. DOI:10.1016/j.jisa.2021.102852.

18. Padma J., Ravishankar N., Raju M.B., Ravi N. (2022) Smart Algorithmsto Secure Web Based Applications from SQL Injection Attacks. *Xi'an Dianzi Keji Daxue Xuebao/Journal of Xidian University*. № 16. pp. 298 - 304. DOI:10.37896/jxu16.2/026.

19. Hadabi A., Elsamani E., Abdallah A., Elhabob R. (2022) An Efficient Model to Detect and Prevent SQL Injection Attack. *Journal of Karary University for Engineering and Science*. DOI:10.54388/jkues.v2i1.141.

References

1. Pedro R., Castro D., Carreira P., Santos N. (2023) From Prompt Injections to SQL Injection Attacks: How Protected is Your LLM-Integrated Web Application? *arXiv*, available at https://doi.org/10.48550/arXiv.2308.01990

2. Sommervoll Å., Erdődi L., Zennaro F. (2023) Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents. *International Journal of Information Security*, vol. 23, available at https://doi.org/10.1007/s10207-023-00738-3

3. Kavetskyi, M., Sievierinov, O., Gvozdov, R., & Smirnov, A. (2024) Using machine learning to classify DOS/DDOS attacks. Radiotekhnika, 2(217), 55–63. https://doi.org/10.30837/rt.2024.2.217.04

4. Asieieva, L. A. (2023) Management of enterprise information security using machine learning and fuzzy logic, doctoral dissertation, Cybersecurity, State University of Information and Communication Technologies, Kyiv, Ukraine.

5. Ivanishchenko Yevhen., Sablina Milana, & Kravchuk Kateryna. (2021). Vykorystannia mashynnoho navchannia v kiberbezpetsi.

Elektronne fakhove naukove vydannia «Kiberbezpeka: osvita, nauka, tekhnika», 4(12), 132–142, available at: https://doi.org/10.28925/2663-4023.2021.12.132142

6. Banakh, R., Piskozub, A., & Opirskyy, I. (2023). Devising a method for detecting “evil twin” attacks on ieee 802.11 networks (wi-fi) with knn classification model. Eastern-European Journal of Enterprise Technologies, 123(9), available at: https://doi.org/10.15587/1729-4061.2023.282131

7. Hacham S., UÇan O. (2023) Detection of Malicious SQL Injections Using SVM and KNN Algorithms. *7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, Istanbul, Turkiye, pp. 1-5. DOI:10.1109/ISAS60782.2023.10391560

8. Nikhitha M., JabbarM.A. (2019) K Nearest Neighbor Based Model for Intrusion Detection System. *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8(2), DOI:10.35940/ijrte.B2458.078219

9. Shah S., Krishna V., Kartheek V., Gururaj R. (2023) A Clustering-Based Approach for Effective Prevention of SQL Injections. *Springer Nature Singapore*, DOI:10.1007/978-981-99-3569-7_29

10. Zhang B., Ren R., Liu J., Jiang M., Ren J., Li J. (2024) SQLPsdem: A Proxy-based Mechanism towards Detecting, Locating and Preventing Second-Order SQL Injections. *IEEE Transactions on Software Engineering*, vol. 50, no. 7, pp. 1807-1826,. DOI:10.1109/TSE.2024.3400404.

11. Kukutla T. (2023) A Comprehensive Guide to SQL Injection Prevention. *International Center for AI and Cyber Security Research and Innovations (CCRI)*, available at: https://insights2techinfo.com/wp-content/uploads/2023/11/A-Comprehensive-Guide-to-SQL-Injection-Prevention-1-6.pdf

12. Sharma, V., Kumar, S. (2023) Multi-input MLP and LSTM-Based Neural Network Model for SQL Injection Detection. *Computer Vision and Robotics*. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-19-7892-0_35

13. Umar K., Bakar A., Zulzalil H., Admodisastro N., Abdullah M. (2016) SQL Injection Attack Roadmap and Fusion. *Indian Journal of Science and Technology*, vol. 9, DOI: 10.17485/ijst/2016/v9i28/97810

14. Kulsoom F., Narejo S., Mehmood Z., Chaudhry H., Butt A., Bashir A. (2022) A Review of Machine Learning-based Human Activity Recognition for Diverse Applications. *Neural Computing and Applications*. № 34, DOI:10.1007/s00521-022-07665-9

15. Silvestre A., Medeiros I., Mordido A. (2024) Towards a SQL Injection Vulnerability Detector Based on Session Types. *LASIGE, Departamento de Informatica, Faculdade de Ci ´ encias*, Universidade de Lisboa, Portugal, pp. 711-718. DOI:10.5220/0012732500003687

16. Crespo-Martínez I., Vega A., Guerrero-Higueras Á., Riego V., Álvarez-Aparicio C., Fernández C. (2022) SQL Injection Attack Detection in Network Flow Data. *Computers & Security*, vol. 127. 103093, DOI:10.1016/j.cose.2023.103093

17. Kasım Ö. (2021) An ensemble classification-based approach to detect attack level of SQL injections. *Journal of Information Security and Applications*, № 59. 102852. DOI:10.1016/j.jisa.2021.102852

18. Padma J., Ravishankar N., Raju M.B., Ravi N. (2022) Smart Algorithmsto Secure Web Based Applications from SQL Injection Attacks. *Xi'an Dianzi Keji Daxue Xuebao/Journal of Xidian University*. № 16. pp. 298 - 304. DOI:10.37896/jxu16.2/026

19. Hadabi A., Elsamani E., Abdallah A., Elhabob R. (2022) An Efficient Model to Detect and Prevent SQL Injection Attack. *Journal of Karary University for Engineering and Science*. DOI:10.54388/jkues.v2i1.141